Réseaux de tenseurs holographiques d'Ising

par

Thomas Gobeil

Mémoire présenté au département de physique
en vue de l'obtention du grade de maître des sciences (M.Sc.)

FACULTÉ des SCIENCES
UNIVERSITÉ de SHERBROOKE

Sherbrooke, Québec, Canada, 16 juin 2020

Le 16 juin 2020

le jury a accepté le mémoire de Monsieur Thomas Gobeil dans sa version finale.

Membres du jury

Professeur David Poulin
Directeur de recherche
Département de physique

Dr. Guillaume Duclos-Cianci
Co-directeur de recherche
Département de physique

Professeur Bertrand Reulet
Membre interne
Département de physique

Professeur Alexandre Blais
Président rapporteur
Département de physique

À ma famille et à Marianne,

Sommaire

La recherche de codes de correction d'erreurs est un point critique dans le développement d'un ordinateur quantique. Malheureusement, plusieurs des codes de correction trouvés jusqu'à maintenant n'ont qu'une valeur théorique, n'ayant pas de base dans un système physique qui peut être réalisé en laboratoire. Il est donc désirable de chercher un phénomène physique qui pourrait supporter un code de correction d'erreurs. Le principe holographique émergeant de certaines théories de gravité quantique, qui affirme qu'il existe une correspondance entre une théorie gravitationnelle dans un espace Anti-de Sitter de dimension $(d+1)$ et une théorie de champ conforme sur la frontière à d dimensions de cet espace, est un candidat pour ce genre de phénomène. L'information locale dans la théorie gravitationnelle est représentée non-localement dans la théorie de champ, ouvrant la possibilité d'un code de correction d'erreurs. Un qubit logique est encodé dans la théorie gravitationnelle, les qubits physiques devenant des états dans la théorie de champ, soit une chaîne de spins dans la théorie d'Ising dans notre cas, qui est un système physiquement intéressant. Si une partie suffisamment petite des qubits physiques est effacée, le qubit logique peut quand même être reconstruit à partir des qubits physiques restants. Concrètement, cette correspondance est implémentée en utilisant des réseaux de tenseurs hyperboliques. Certains de ces réseaux permettent de représenter l'état appartenant à la théorie de champ conforme tout en possédant une structure interne qui détient les mêmes caractéristiques que l'espace AdS. Si une isométrie entre les tenseurs sur la frontière d'un tel réseau et le tenseur central du réseau existe, un code de correction d'erreurs devrait exister sur le réseau. La création de ces réseaux a été proposée par Jahn et al. en 2017. Ils ont démontré que l'utilisation de portes de parité, une classe de tenseurs introduite par Leslie Valiant en 2002 qui représente des états gaussiens, dans un réseau de tenseurs hyperbolique permet de retrouver un état propre d'un hamiltonien d'Ising à la frontière du réseau. Cependant, l'article qui décrit les réseaux hyperboliques de portes de parité n'offre pas de code explicite pour reproduire ces réseaux, et les explications contenues dans l'article sont parsemées de coquilles et omettent de l'information cruciale. Dans ce mémoire, ces problèmes sont

résolus. Nous offrons d'abord une explication théorique plus accessible en introduisant les tenseurs et leurs réseaux, la correspondance AdS/CFT et le formalisme des portes de parité. Nous présentons également un algorithme permettant de recréer les réseaux discutés dans l'article de Jahn et al. Cette implémentation est générale, permettant de produire un réseau avec un dallage et une profondeur arbitraires tout en imposant les caractéristiques nécessaires à la correspondance AdS/CFT. L'algorithme inclut une fonction qui crée la liste d'adjacence du réseau de tenseurs ainsi que les fonctions requises pour manipuler les portes de parité et effectuer le passage de la liste d'adjacence au réseau de tenseurs contracté. De là, nous donnons également le code nécessaire pour calculer la matrice de covariance de l'état résultant, de laquelle tous les corrélateurs peuvent être extraits. Avec cet algorithme nous sommes capables de recréer les figures clés dans l'article de Jahn et al., confirmant la validité du code.

quantique, holographie, réseau, tenseur, hyperbolique, AdS/CFT, Ising

Remerciements

Je voudrais remercier mon superviseur, Pr. David Poulin, pour son aide, sa patience et sa compréhension lors de ce projet. Il a été et demeure une source inébranlable d'encadrement et de savoir, et je suis reconnaissant d'avoir pu travailler avec lui. Je veux également remercier Guillaume Duclos-Cianci pour avoir pris la relève et réussit à amener ce projet jusqu'à complétion malgré les circonstances difficiles. À Glen Evenbly et à Alexander Jahn je souhaite transmettre ma gratitude pour avoir pris le temps de m'expliquer leurs codes et de répondre à mes incessantes questions. Je veux finalement remercier tous mes collègues qui ont pris le temps de lire mon mémoire et de me donner des commentaires constructifs.

Table des matières

Table des figures

Introduction

La réalité d'un ordinateur quantique devient de plus en plus imminente d'année en année. Cependant, il reste encore une pléthore d'embûches avant d'arriver à un outil à la fois pratique et capable d'accomplir toutes les tâches qu'on attend d'un tel ordinateur. Un obstacle qui subsiste est le problème de la correction d'erreur. Dans un ordinateur classique, il est possible de facilement observer et corriger la grande majorité des erreurs qui apparaissent, et les occurrences d'erreur ne sont pas très fréquentes. Un ordinateur quantique, au contraire, ne profite pas du comportement classique qui permet de mesurer les erreurs et de les corriger facilement, et la sensibilité à l'environnement du domaine quantique fait en sorte que des erreurs se produisent fréquemment. Il faut donc développer des codes qui tolèrent une certaine quantité d'erreurs, ainsi que d'autres qui permettent de corriger le circuit après l'apparition d'une erreur.

Une multitude de codes existent déjà, chacun ayant certains avantages comparativement aux autres ([3]-[12]). Il existe même un code torique en quatre dimensions qui devient plus tolérant aux erreurs quand la taille du code augmente [13]. Le problème derrière ce genre de code est bien sûr le fait qu'il n'est pas physiquement possible, étant donné qu'il n'y a que trois dimensions spatiales qui nous sont accessibles. Une contrainte supplémentaire émerge de cette recherche : nous devons trouver un code de correction d'erreur qui peut être réalisé aisément en laboratoire. Pour ce faire, on tourne notre attention vers un outil mathématique qui est largement utilisé à la fois en matière condensée ([14]-[19]) et en information quantique ([20]-[26]), soit les réseaux de tenseurs.

Dans ce mémoire, nous allons explorer une nouvelle approche à certains réseaux de tenseurs qui montrent un grand potentiel pour la correction d'erreur. Pour ce faire, nous allons en premier introduire certaines notions, tels les tenseurs, qui seront nécessaires à la compréhension de cette approche. Ces idées préliminaires, qui seront discutées juste en survol dans l'introduction, sont sondées plus en profondeur dans la théorie. Pour l'instant nous supposons que le lecteur a déjà des bases en mécanique quantique ainsi qu'en algèbre

linéaire.

Un tenseur est un concept mathématique qui généralise l'idée d'une application linéaire [27]. Plus concrètement, un tenseur est une fonction de multiples paramètres qui est linéaire pour chacun de ces paramètres. Certains exemples communs de tenseurs sont les vecteurs et les matrices, qui sont des tenseurs de rang 1 et de rang 2 respectivement. Un réseau de tenseurs est simplement un ensemble de tenseurs qui sont reliés par des contractions, qui sont équivalentes à un produit de matrices. Les réseaux de tenseurs ont été développés à l'origine pour traiter des problèmes de systèmes à N corps, et peuvent représenter efficacement des fonctions d'onde dans un espace de Hilbert exponentiellement large [16]. L'application de cet outil s'est rapidement répandue plus loin que les bornes de ce domaine de la physique, et éventuellement il fût proposé qu'un réseau de tenseurs pouvait réaliser la correspondance AdS/CFT. Les abréviations ici dénotent l'espace Anti-de Sitter (AdS), une version théorique de l'espace-temps décrite par une métrique hyperbolique, et une théorie de champ conforme, soit une théorie qui décrit le comportement et les interactions des particules fondamentales.

La correspondance AdS/CFT relie la gravité avec la mécanique quantique en démontrant que l'information contenue dans l'espace Anti-de Sitter de dimension $(d+1)$ peut être retrouvée sur une théorie de champ conforme correspondante de dimension d. Ce principe est nommé le principe holographique. Récemment, il a été postulé que cette caractéristique de la correspondance AdS/CFT peut être formulée dans le langage de la correction d'erreur [28]. L'information qui doit être conservée serait emmagasinée dans l'espace AdS, qui contient les qubits logiques. Les qubits physiques, eux, sont identifiés avec la théorie de champ conforme sous la forme d'un système de spins, par exemple. La structure d'intrication de la théorie de champ protégerait la perte d'information dans le volume AdS quand une partie suffisamment petite des qubits sur la frontière CFT serait effacée, menant à un code de correction d'erreur qui utilise l'information préservée pour reconstruire le sous-système erroné [29].

Initialement, certains auteurs ont proposés qu'un MERA (Multi-scale Entanglement Renormalization Ansatz), un type de réseau de tenseurs avec des caractéristiques propres à l'holographie, était apte à représenter la correspondance AdS/CFT. Cependant, ce type de réseau possède une orientation préférentielle qui ne permet pas une représentation fidèle de la correspondance, et qui empêche les stratégies de correction d'erreur associées avec celle-ci. La prochaine proposition parut en 2015 par Pastawski et al [29]. Dans cet article, ils proposent qu'un réseau composé de *tenseurs parfaits*, c'est-à-dire des tenseurs qui sont également des isométries, permet de réaliser la correction d'erreur. Bien que leur hypothèse ait été majoritairement validée, il est possible de démontrer que la théorie de champ conforme créée en frontière de leur réseau n'est pas physiquement intéressante, car les corrélations

entre les sites ne diminuent pas de manière algébrique, une facette importante de théories de champ comme celle d'Ising [2]. Une alternative proposée par [2] est de trouver un réseau de tenseurs dans lequel certaines combinaisons de tenseurs peuvent être équivalentes à un tenseur parfait. La relaxation de cette contrainte permet également de construire un réseau avec des corrélateurs diminuant de façon exponentielle avec la distance entre les sites. Ces réseaux doivent répondre à quatre critères :

(1) Ils sont bâtis d'un dallage uniforme de l'espace hyperbolique.

(2) Ils peuvent être calculés de manière efficace.

(3) Ils ont des paramètres variationels libres.

(4) Ils encodent des états quantiques avec une diminution algébrique des fonctions de corrélation à deux points.

Il n'est pas encore clair quels états peuvent être représentés sur la surface de ces codes, ce qui demeure la plus grande incertitude sur leur validité. Un autre code émergeant du travail sur les tenseurs parfaits fût un article par Jahn, Gluza, Eisert et Pastawski [30]. Dans cet article, ils approchent les réseaux hyperboliques avec un type de tenseur dénommé *porte de parité* ("matchgate"). Ce tenseur, initialement développé par Valiant [31], et généralisé par Bravyi [32], permet la création d'un réseau de tenseurs hyperbolique avec une théorie de champ d'Ising sur la frontière du réseau. Encore plus prometteur, Jahn et al. ont démontré que le réseau de tenseurs parfaits était un cas spécifique d'un réseau de portes de parité. La question demeure si nous sommes capables de trouver des agencements de tenseurs dans ces réseaux qui permettent de créer des tenseurs parfaits sans éliminer les fonctions de corrélations non-triviales.

Le but de ce travail est d'introduire les réseaux de portes de parité et le formalisme associé, ainsi que de fournir un programme capable de créer des réseaux de portes de parité et de calculer de manière efficace les corrélateurs. Dans le premier chapitre, nous introduisons d'abords les tenseurs en général et les opérations couramment utilisées pour les manipuler. Nous parlons également plus en profondeur de la correspondance AdS/CFT ainsi que de l'holographie et des réseaux hyperboliques. Par la suite, nous donnons le formalisme des opérateurs de Grassmann nécessaire à la compréhension et au traitement mathématique des portes de parité, qui sont introduits plus formellement dans la section suivante. Le deuxième chapitre de ce travail donne le code que nous avons développé afin de traiter des réseaux de portes de parité. Le code est décomposé en plusieurs parties, qui sont introduites une à la fois avec leur explication respective. Nous commençons par l'algorithme pour créer la liste d'adjacence, qui détermine la position de chaque tenseur dans le réseau relatif aux

autres tenseurs du même réseau, introduisant en même temps les sous-fonctions requises pour le faire. Deux algorithmes seront explicitement présentés et comparés. Ensuite sera donné le code pour effectuer la contraction du réseau. Ce code introduira également les trois opérations de base pour la manipulation d'un réseau de tenseurs : la permutation cyclique, la contraction et l'auto-contraction. La dernière section présente l'algorithme pour convertir le réseau résultant en une matrice de covariance, de laquelle les corrélateurs peuvent être extraits.

Chapitre 1

Théorie

1.1 Réseaux de tenseurs

Les tenseurs sont des objets mathématiques pratiques et versatiles, utilisés dans la résolution de systèmes quantiques ([33], [34]), l'intelligence artificielle ([22]-[35]), la correction d'erreur ([28], [36]-[37]), et plusieurs autres domaines. La section suivante servira d'introduction aux tenseurs et à leurs réseaux ainsi qu'au formalisme associé. Nous parlerons également plus en détails d'un type de réseau de tenseurs nommé MERA et de son lien avec l'holographie. Commençons par donner une définition formelle d'un tenseur.

Définition 1.1.1. Un tenseur de rang n dans un espace de d dimensions est un objet mathématique avec d^n composantes spécifiées par n indices. Chaque indice recouvre le nombre de dimensions de l'espace. Pour un tenseur T de rang 3 dans un espace à deux dimensions (x étant binaire), un élément de tenseur pour $x = \{x_1, x_2, x_3\}$ serait spécifié par

$$T(x_1, x_2, x_3) \quad \text{ou} \quad T_{x_1 x_2 x_3}, \tag{1.1}$$

avec $T(x)$ et T_x étant des notations équivalentes.

Certains tenseurs sont communs et fréquemment utilisés dans la littérature sous d'autres noms. Un tenseur de rang 0 est un scalaire, un tenseur de rang 1 est un vecteur, et un tenseur de rang 2 est une matrice.

Les tenseurs sont utilisés afin de décomposer un système quantique complexe, ce qui permet de simplifier de sa résolution. La clé de cette simplification est une opération de décomposition qui divise le tenseur en un réseau de tenseurs de rang inférieur. Un réseau

de tenseurs est un objet composé de L tenseurs, chacun possédant un rang et une dimension propre. Le rang total du réseau est déterminé par le nombre d'indices non-contractés dans le réseau. Nous allons discuter de la transition entre le tenseur et le réseau dans la section sur le formalisme, mais en premier nous discuterons de la représentation graphique des tenseurs, qui est beaucoup plus intuitive.

1.1.1 Représentation graphique

Dans le reste du mémoire, nous allons représenter les tenseurs graphiquement. Un tenseur aura la forme d'un polygone, son rang correspondant au nombre de côtés du polygone. Afin d'illustrer certaines propriétés des tenseurs, nous allons ajouter une arête à chaque côté qui correspondra à l'indice de ce côté, comme l'est illustré dans la figure 1.1. Il y aura quelques exceptions à ceci. Dans le cas où un tenseur est d'un rang tel qu'il serait inefficace de le représenter sous forme de polygone, nous allons l'illustrer avec un rectangle au coins arrondis et le nombre d'arêtes correspondant à son rang. Pour les tenseurs de degré inférieur à 3, ils seront représentés par un cercle avec le nombre d'arêtes correspondant à leur rang.

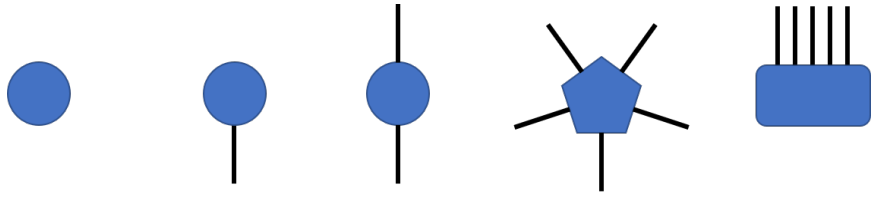
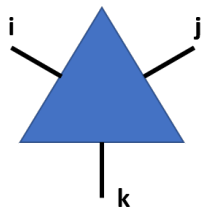


Figure 1.1 Des tenseurs de différents rangs. De droite à gauche, nous avons un tenseur de rang 0 (un scalaire), un tenseur de rang 1 (un vecteur), un tenseur de rang 2 (une matrice), un tenseur de rang 5 et une autre visualisation d'un tenseur de rang 5.

Considérant que nous allons travailler avec des fermions dans une théorie de champ d'Ising, nous allons identifier chaque arête à un spin fermionique. Dans ce cas, la dimension de l'espace sera de 2. Nous effectuons ainsi un passage dans l'interprétation des tenseurs d'une application linéaire à des amplitudes d'un état quantique. Un tenseur représentant un état à trois spins $i, j, k = \{0, 1\}$ est désigné par



où 0 correspond à un spin "down" et 1 désigne un spin "up". Donc, le tenseur propre à l'état $\frac{1}{2}(|000\rangle + |101\rangle + |110\rangle + |011\rangle)$ aura les éléments

$$T(000) = \frac{1}{2} \qquad T(001) = 0 \qquad T(010) = 0 \qquad T(100) = 0$$
$$T(101) = \frac{1}{2} \qquad T(110) = \frac{1}{2} \qquad T(011) = \frac{1}{2} \qquad T(111) = 0$$

Chaque arête correspond à un site, avec l'indice de l'arête dénotant le spin de ce site.

Dans certains cas, l'orientation de l'arête dénotant un indice indique si cet indice est covariant au contravariant selon si l'arête est vers le haut ou vers le bas (gauche et droite sont également utilisées). Cependant, cette convention n'a pas de sens bien défini dans les réseaux que nous allons utiliser, donc elle ne sera pas appliquée ici.

1.1.2 Formalisme

Comme pour les scalaires, les vecteurs, et les matrices, les tenseurs de rang plus élevé sont sujets à une multitude d'opérations possibles. Quelques opérations, introduites ici, sont considérées essentielles et apparaissent de manière universelle dans les algorithmes traitant avec des tenseurs dans une forme ou une autre. Bien que nous ne couvrons que la théorie mathématique dans cette introduction, ces opérations peuvent être implémentées aisément dans Julia, Matlab, et d'autres programmes de haut niveau. Nous allons couvrir quatre opérations, soit le remodelage, la contraction, la permutation et la décomposition. Bien que plusieurs articles sur les tenseurs donnent habituellement une base algorithmique sur l'application pratique de ces opérations, nous allons éviter de le faire dû à la nature particulière des tenseurs que nous traitons.

Remodelage

Le remodelage est le changement du rang d'un tenseur en combinant plusieurs indices en un, ou en divisant un indice en plusieurs. Par exemple, une matrice peut être remodelée en un vecteur

$$T_{ij} \to T_k.$$

Pour ce faire, nous devons imposer un ordre à ce remodelage. Cet ordre dicte quelle variable est incrémentée en premier dans la variable résultante après le remodelage. Étant donné que nous utilisons Matlab comme outil de programmation, nous suivons l'ordre donné par le logiciel. Le dernier indice sera donc le premier indice incrémenté, donnant le tableau suivant

i	1	1	$\cdots$	1	2	$\cdots$	2	$\cdots$	n_i
j	1	2	$\cdots$	n_j	1	$\cdots$	n_j	$\cdots$	n_j
k	1	2	$\cdots$	n_j	n_j+1	$\cdots$	$2n_j$	$\cdots$	n_in_j

où n_i et n_j sont les dimensions des indices i et j respectivement. La formule pour trouver l'indice k selon les indices i et j est $(i-1)n_j + j$. Graphiquement, ce remodelage correspond au regroupement de deux indices (voir la figure 1.2). La séparation d'un indice se fait en suivant le protocole inverse au regroupement.

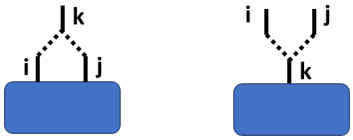


FIGURE 1.2 À gauche : la combinaison de deux indices en un, remodelant une matrice en un vecteur. À droite : la décomposition d'un indice en deux, effectuant le processus inverse.

Contraction

La contraction de deux tenseurs se résume analytiquement à la contraction sur un indice partagé. Prenons deux tenseurs de rang 3, soit A et B, qui seront contractés dans un tenseur C de rang 4.

$$C_{ijlm} = \sum_k A_{ijk} B_{klm} \tag{1.2}$$

Cette contraction peut être représentée graphiquement en rejoignant les deux arêtes correspondant aux indices contractés, comme dans la figure 1.3, qui démontre le même principe avec des tenseurs de rang 1 et 2.

Le coût de calcul d'une contraction typique est normalement donné par le nombre d'arêtes impliquées dans la contraction, à la fois les arêtes libres et celles contractées. Prenons

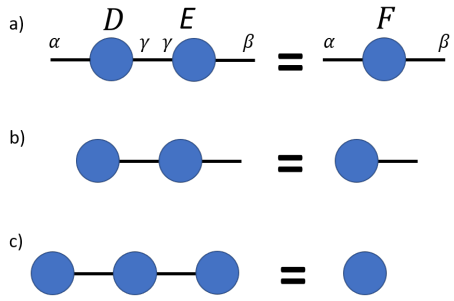


FIGURE 1.3 La contraction des matrice D et E sur l'indice partagé γ, résultant en la matrice F (a). Suivant est une contraction d'un vecteur avec une matrice résultant en un vecteur (b), puis le calcul d'un corrélateur à deux points résultant à un scalaire (c).

la figure 1.4 comme exemple, et attribuons une dimension χ à chaque arête. Afin de savoir le coût total de la contraction, nous dessinons une boîte autour des deux tenseurs contractés ensemble. On compte ensuite le nombre d'arêtes incluses dans l'enceinte de la boîte, incluant celle qui lie les deux tenseurs. Dénotons ce nombre s. Le coût total de la contraction sera de $O(\chi^s)$.

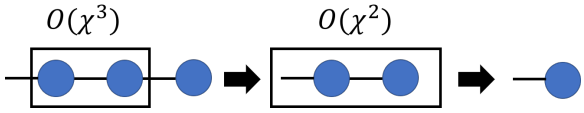


FIGURE 1.4 La contraction de deux matrices et d'un vecteur. Lors de la première contraction, il y a une branche contractée et deux branches coupées par la boîte, donnant un nombre de trois branches et donc un coût de $O(\chi^3)$. La deuxième contraction prend en compte deux arêtes, donnant $O(\chi^2)$.

Bien que la contraction entre deux tenseurs soit triviale, la contraction d'un réseau est un problème complexe [38]. On montre dans la figure 1.5 deux manières différentes de contracter le même réseau de tenseurs. Une de ces méthodes limite le coût de contraction à $O(\chi^5)$, tandis que l'autre a une étape avec un coût de $O(\chi^6)$, ce qui peut être une énorme différence pour un espace de dimension élevée.

Décomposition

Il est possible de subdiviser un tenseur. Pour mieux illustrer cette décomposition, nous allons remodeler le tenseur de rang arbitraire en une matrice. La décomposition d'un tenseur reviendra donc à une décomposition de matrice. Une décomposition sépare le tenseur en deux, chacun avec une arête, comme l'est démontré dans la figure 1.6. Répéter une décomposition plusieurs fois, chaque fois isolant par remodelage un des indices, mène à

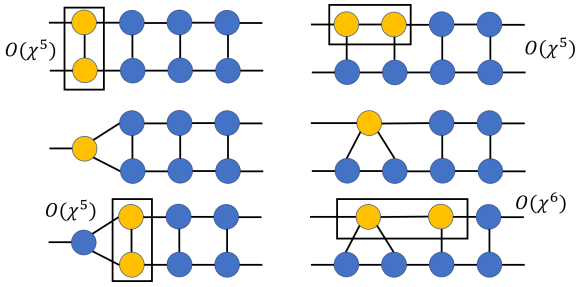


FIGURE 1.5 Deux stratégies de contraction différentes. La première, à gauche allant de haut en bas, limite le coût de contraction à $O(\chi^5)$ en répétant la même contraction le long de la chaîne. À droite, les tenseurs sur le haut de la chaîne sont contractés ensemble, augmentant le coût par un facteur de χ à chaque étape.

la création d'un réseau de tenseurs, avec chaque tenseur maintenant associé à un site plutôt qu'à l'état entier. La décomposition du tenseur se fait de deux manières, soit la décomposition en valeurs propres et la décomposition en valeurs singulières (DVS).

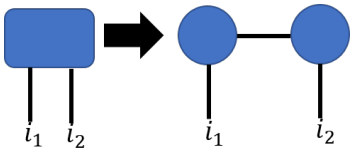


FIGURE 1.6 La décomposition d'un tenseur de rang 2 en deux tenseurs de rang 2, avec les nouveaux indices contractés ensemble. Chaque tenseur possède un des indices du tenseur original.

La décomposition en valeurs propres est simplement la diagonalisation d'une matrice, la décomposant en la forme

$$A_{ij} = \sum_{mn} U_{im} \rho_{mn} U^{\dagger}_{nj} \tag{1.3}$$

où ρ est la matrice diagonale contenant les valeurs propres [39]. Souvent ce tenseur diagonal sera absorbé dans un des deux tenseurs avec un indice libre. Comme on peut le voir, les deux tenseurs décrits par $U^{\dagger}$ et U gardent un des indices originaux.

La décomposition en valeurs singulières (DVS) est une opération générale applicable à n'importe quelle transformation linéaire et permet également d'obtenir une matrice diagonale avec des valeurs propres. D'ailleurs, si la transformation est normale la DVS se réduit à la diagonalisation, mais pour un cas général elle décompose la transformation linéaire représentée par la matrice T en trois matrices S, U et D. Les matrices U et $U^{\dagger}$, qui sont des matrices unitaires, sont remplacées par les matrices S et $V^{\dagger}$ respectivement, qui sont des isométries. Des isométries sont des matrices qui ont la propriété d'avoir un produit qui

correspond à l'identité dans un seul sens, soit

$$S^\dagger S = I, \quad SS^\dagger \neq I.$$

La décomposition finale est donnée par

$$A_{ij} = \sum_{mn} S_{im} D_{mn} V^\dagger_{nj} \tag{1.4}$$

où D est diagonale et de dimension $(min(n_i, n_j), min(n_i, n_j))$, avec n_x la taille de l'indice x. Les dimensions de S seront donc $(n_i, min(n_i, n_j))$ et celles de $V^\dagger$ seront $(min(n_i, n_j), n_j)$ [27].

Cette opération est exacte, mais d'un premier coup d'oeil il semble y avoir une perte d'information. La quantité d'information apparemment perdue se visualise assez aisément en regardant la décomposition d'un tenseur représentant N sites fermioniques en N tenseurs, chacun représentant un seul site, dans la figure 1.7.

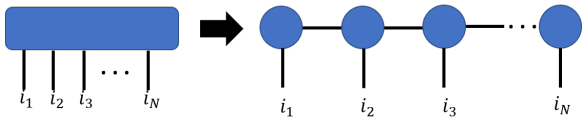


FIGURE 1.7 Comme la décomposition dans la figure précédente, appliquée sur un tenseur de rang N. Chaque nouveau tenseur a un nombre fixe de valeurs possibles sur son indice libre, donnant une taille $O(N)$ à la chaîne décomposée comparativement au tenseur initial qui à une taille de $O(2^N)$.

Si on regarde les coefficients requis pour exprimer l'état, on voit que dans le tenseur initial nous avions besoin de $O(2^N)$ coefficients, tandis que dans la chaîne nous n'avons besoin que de $O(N)$ coefficients. L'information n'est toutefois pas nécessairement perdue, donc où se retrouve-t-elle ? Elle est cachée dans les indices contractés de la chaîne, dans les liens horizontaux. Ces liens entre les tenseurs possèdent l'information sur l'intrication entre les sites, et une décomposition exacte demande que la taille de ces liens soit exponentielle [38]. Il est cependant possible de tronquer les plus petites valeurs de la matrice diagonale, ce qui est équivalent à réduire la taille des indices internes.

Bien que ceci soit utile pour représenter efficacement certains états, il est important de regarder quelle information est contenue dans ces indices internes. La réponse est que ces indices représentent l'intrication entre les différents sites représentés par les indices ouverts. Dans la figure 1.8, on voit qu'un réseau de deux tenseurs peut être simplifié selon l'intrication entre les deux. Désignons par χ la dimension d'un indice interne dans un tenseur à deux sites. Dans le cas d'un état non-intriqué, l'arête connectant les deux ne contient pas d'information

($\chi = 1$) et peut être enlevée, tandis que pour un état où l'intrication est maximale, comme un état de Bell, la connaissance d'un site donne la connaissance de l'autre, et les deux tenseurs se réduisent à l'identité, qui est simplement une arête en deux dimensions [27].

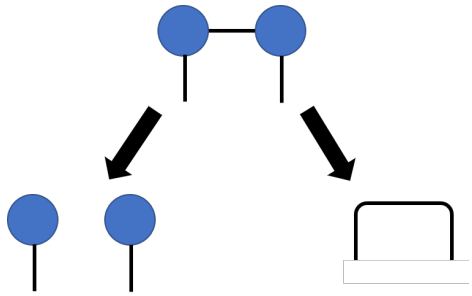


FIGURE 1.8 Un réseau de tenseurs peut être simplifié selon la dimension χ de l'arête reliant les deux tenseurs. Si la dimension est de 0, les deux états ne sont pas intriqués et peuvent être séparés en deux tenseurs. Si l'intrication est maximale, la valeur du premier indice spécifie la valeur du deuxième. Le résultat est donc l'identité, soit une simple arête.

Nous allons donner un exemple concret et intuitif pour ce genre de décomposition. Nous allons prendre l'état à trois qubits

$$|\Psi\rangle = \frac{1}{2}(|001\rangle + |010\rangle - |101\rangle - |110\rangle). \tag{1.5}$$

Cet état peut certainement être décomposé sans avoir recours à un logiciel quelconque, mais nous allons néanmoins faire la démonstration ici. Initialement, cet état est un tenseur de rang 3. Pour faire la première décomposition en valeur singulière, nous effectuons un remodelage des indices 2 et 3 ensemble (le deuxième et troisième qubit). Cela résulte en la matrice

$$M_\Psi = \frac{1}{2}\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & -1 & -1 & 0 \end{pmatrix}. \tag{1.6}$$

Effectuant une DVS sur cette matrice, on retrouve les trois matrices

$$S = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}, \quad D = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad V^\dagger = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 & 1 & 1 & 0 \\ \sqrt{2} & 0 & 0 & \sqrt{2} \end{pmatrix}. \tag{1.7}$$

En remarquant que la matrice D ne possède qu'une valeur, nous voyons qu'il existe un

produit de deux états. Regroupant S et D, on retrouve la matrice

$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 \\ -1 & 0 \end{pmatrix}, \tag{1.8}$$

Ce qui révèle l'état $|\phi_1\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$. Pour continuer la décomposition du tenseur, nous regardons la matrice $V^\dagger$. Nous voyons intuitivement qu'elle peut également être simplifiée en un vecteur indépendant, puisqu'elle faisait partie de l'état non-intriqué. Remarquant que seule la première rangée serait conservée dans le produit de matrices, nous retrouvons le vecteur

$$M_{\phi_{2,3}} = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 & 1 & 1 & 0 \end{pmatrix}. \tag{1.9}$$

Un remodelage de ce vecteur en matrice pour le préparer à une dernière DVS donne

$$M_{\phi_{2,3}} = \frac{1}{\sqrt{2}}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{1.10}$$

qui est σ_x. L'état est donc un état de Bell, maximalement intriqué. Le résultat de la décomposition est illustré dans la figure 1.9.

Donc qu'advient-il donc de l'habileté des tenseurs à recouvrir la totalité de l'espace de Hilbert ? Pour répondre à cette question, regardons le recouvrement de l'espace de Hilbert en fonction de χ, qui sera la dimension de l'espace du lien entre les tenseurs dans la chaîne. Nous allons illustrer ce recouvrement de manière qualitative avec la figure 1.10. Comme on le voit, quand χ augmente nous pouvons représenter une plus grande section de l'espace de Hilbert, et quand $\chi \to \exp(N)$ on retrouve la totalité de l'espace. Mais nous sommes préoccupés par l'étude d'états physiques, plus précisément d'états qui respectent la loi de l'aire ("area law"), qui sont recouverts avec un χ relativement petit [38]. Nous pouvons donc limiter la taille de χ sans perdre trop de précision dans nos calculs, donc la troncature de D n'est pas néfaste si nous enlevons les plus petites valeurs. La précision peut ensuite être choisie en mettant un seuil aux valeurs tronquées.

La DVS est donc un bon choix pour décomposer tous les types de tenseurs. En effet, si nous voulons avoir une chaîne de tenseurs pour cinq sites, la première décomposition demande de remodeler le tenseur initial en un tenseur à deux sites en regroupant quatre indices en un. La matrice résultante ne sera pas carrée, et donc la décomposition en valeur propre sera impossible, laissant la DVS comme la seule option.

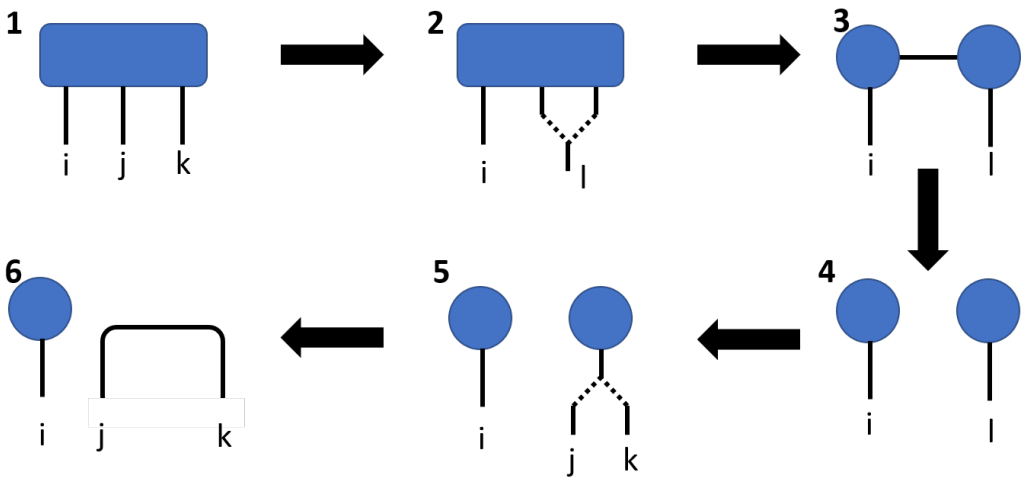


FIGURE 1.9 Décomposition d'un état à trois qubits (1). Nous commençons par un remodelage dans l'étape 2 des indices j et k dans l'indice l, suivi d'une première DVS dans l'étape 3. Le rang de D étant 1, nous concluons que les deux états ne sont pas intriqués lors de l'étape 4, séparant les indices i et l sur deux tenseurs. Un remodelage du tenseur à droite de l'indice l en les indices j et k au cours de l'étape 5 donne σ_x, un état maximalement intriqué, concluant la décomposition.

Permutation

Dans plusieurs algorithmes, certaines opérations ne sont faisables que sur deux indices adjacents. La permutation permet de réorganiser les indices afin de permettre l'implémentation de ces opérations. Le remodelage en est une qui nécessite souvent une série de permutations, mais dans le cas des portes de parité c'est la contraction qui sera la cause première de permutations cycliques des indices d'un tenseur.

La permutation est une opération normalement coûteuse en pratique, car elle requiert la reconstruction du tenseur, bien qu'elle est aisément réalisée analytiquement et graphiquement en changeant l'étiquette des arêtes pour refléter le nouvel indice [40]. Dans le cas de la permutation cyclique d'une porte de parité, cependant, ce coût peut être évité facilement en suivant l'algorithme donné dans le texte.

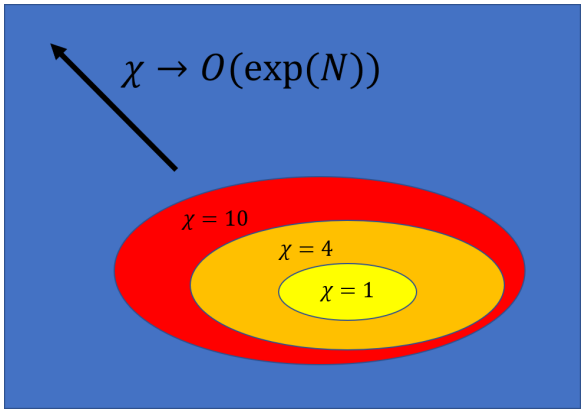


FIGURE 1.10 L'espace de Hilbert est ici représenté par le rectangle bleu. L'évolution du recouvrement par l'augmentation de la taille de χ est dénotée par les ovales de différentes couleurs. Cette représentation est qualitative.

1.2 La correspondance AdS/CFT et MERA

Avant de parler du réseau de tenseurs central à ce mémoire, regardons d'abord le concept qui a motivé la recherche sur les portes de parité et la correction d'erreur. Ce concept est la correspondance AdS/CFT, dont nous avons parlé en introduction. Afin de mieux expliquer ce concept, regardons d'abords l'espace Anti-de Sitter. Cet espace est une des métriques les plus fréquemment utilisées en relativité générale, car il s'agit d'un modèle jouet pour des métriques plus complexes, particulièrement dans l'étude de l'intersection entre la relativité et la mécanique quantique, spécifiquement la correspondance AdS/CFT [41]. Cette notion implique qu'il existe une dualité entre une théorie de champ conforme et un espace-temps décrit par la métrique Anti-de Sitter, et que les équations fondamentales de la théorie de champ peuvent être dérivées à partir de la métrique de l'espace-temps. Un exemple de ceci est la correspondance entre l'espace $AdS_5 \times S^5$ et la théorie N = 4 de Yang-Mills. Cette dualité est réalisée par le principe holographique, qui dit qu'un espace de dimension $(d+1)$ peut être projeté sur la frontière de cet espace, qui est de dimension d. C'est sur cette frontière que la théorie de champ apparaît. Ceci peut être relativement difficile à imaginer, donc nous allons représenter cela sous forme graphique en effectuant ce qui s'appelle une compactification de l'espace AdS. Une compactification revient à rendre fini un espace infini en repliant une de ses dimensions sur elle-même et en ajoutant un point à l'infini. Quand cette opération est effectuée sur l'espace Anti-de Sitter, on retrouve que son algèbre correspond à l'algèbre de Poincarré. Nous allons donc représenter l'espace AdS comme un cercle de Poincarré, et pour l'instant nous allons mettre un dallage hyperbolique afin d'illustrer la courbure négative de l'espace, qui est une des caractéristiques fondamentales de l'espace AdS [42]. Cette illustration se retrouve à la figure 1.11. Afin de faire correspondre notre futur réseau

de tenseurs à cette géométrie, on imagine placer un tenseur sur chaque face avec un rang égal au nombre d'arêtes de la face. L'état résultant, se trouvant sur la frontière de l'espace AdS, sera un état appartenant à une théorie de champ d'Ising.

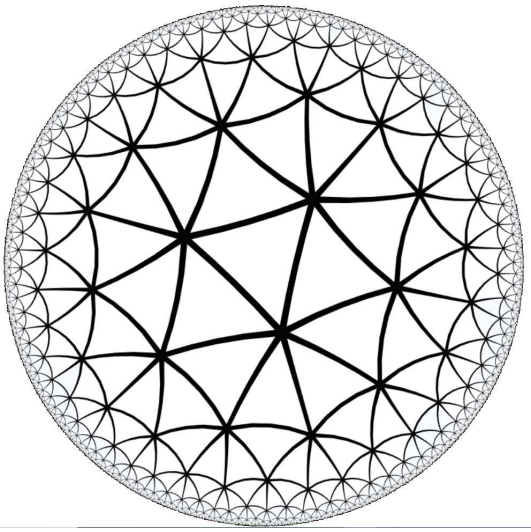


FIGURE 1.11 Un des dallages possibles du cercle de Poincarré. Le dallage utilisé est le dallage {3,7}, chaque tenseur le composant étant donc un tenseur de rang 3, et nous pouvons retrouver 7 tenseurs autour de chaque sommet dans le dallage. Cette représentation est dans le continuum, et il existe donc un nombre de tenseurs infini entre le tenseur central et la frontière.

Par hyperbolique, nous voulons dire que, pour chaque sommet dans le graphe contenant la géométrie, il y a q polygones avec p côtés qui contiennent ce sommet, avec q et p respectant la relation

$$(p-2)(q-2) > 4. \tag{1.11}$$

Un dallage respectant cette relation est hyperbolique et est dénoté {p,q}. La figure 1.11 montre un dallage {3,7}. Comme nous l'avons brièvement mentionné plus haut, la raison pour laquelle nous utilisons un dallage hyperbolique est pour représenter la courbure négative de l'espace AdS. En fait, si nous calculons la projection de l'espace AdS sur un plan en deux dimensions, on remarque que cette projection serait isomorphe au plan hyperbolique. Une démonstration de ce calcul est fait par Rafael Lopez [43]. Dû à cette géométrie, chaque triangle dans la figure 1.11 possède la même aire et se situe à une distance infinie du périmètre du cercle.

Qu'est-ce l'holographie, vraiment ? L'holographie, aussi appelée dualité holographique, est la représentation d'un objet de dimension $(d+1)$ par un objet de dimension d, comme

un hologramme qui détient l'information complète en trois dimension d'un sujet, tout en étant une projection en 2 dimensions [42].

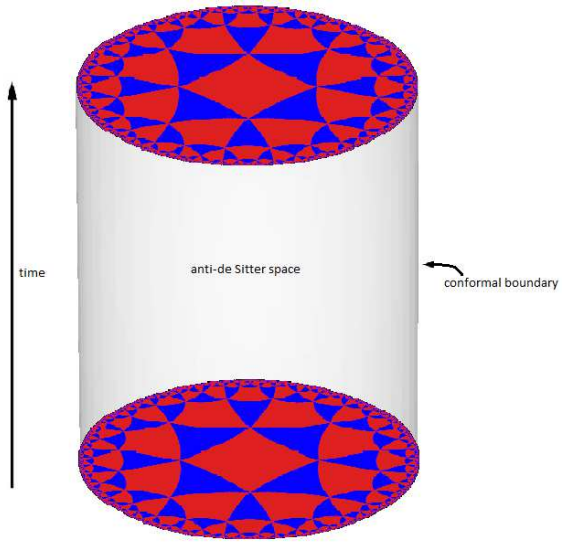


FIGURE 1.12 Nous retrouvons l'espace AdS, qui est composé de plusieurs tranches représentant l'espace spatial. La direction en z est identifiée avec le temps. La frontière du cylindre est où se situe la théorie de champ conforme. Figure tirée de [1].

Dans la correspondance AdS/CFT la géométrie de l'espace Anti-de Sitter est l'objet, et la théorie de champ conforme est l'hologramme. Plus précisément, la frontière de l'espace AdS est l'espace-temps du champ conforme, car la frontière ressemble localement à un espace de Minkowski. La représentation complète de l'espace AdS sera donc donnée par 1.12.

Si la théorie de champ conforme (la frontière du réseau) contient toute l'information disponible dans l'espace AdS (le volume du réseau), il est logique que la connaissance de l'état sur la frontière permet de reconstruire l'état dans le volume. Si nous perdons de l'information sur l'état sur la frontière, nous allons également perdre notre habileté de reconstruire le volume. C'est ici que nous pouvons faire un lien avec la correction d'erreur. La structure d'intrication d'une théorie de champ conforme limite l'impact d'une perte d'information. Il est également possible de reconstruire un opérateur dans le volume à partir de différents opérateurs sur la frontière [29]. Cela revient à une délocalisation de l'information locale dans le volume. Ce concept est illustré dans la figure 1.13, qui montre le cône causal (le même que celui illustré dans la figure 1.14) d'une région $C(A)$ de la frontière. Plus grande est la région perdue, plus loin dans le volume l'impact est répandu. L'information peut ensuite être reconstruite en utilisant le complément de la frontière perdue, tant que la perte n'est pas trop significative [29]. Certains réseaux possèdent des caractéristiques capables de reproduire les traits de la correspondance AdS/CFT, et qui peuvent corriger les

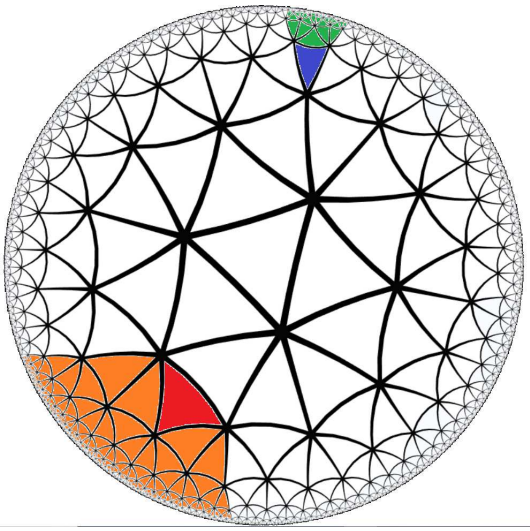


FIGURE 1.13 Une représentation de la délocalisation de l'information à différentes échelles. L'information contenue dans le tenseur en rouge dans le volume peut être reconstruite à partir de l'information contenue sur la frontière du cône orange. La même chose est vraie pour l'information contenue dans le tenseur bleu, qui se retrouve sur la frontière du cône vert. On voit qu'un tenseur plus proche du centre du réseau demande une plus grande surface pour le représenter, voulant dire que l'information est d'autant plus délocalisée.

erreurs de la même manière.

Une classe variationelle de réseaux de tenseur en particulier est importante dans le contexte des articles sur l'holographie et les portes de parité, soit MERA. En particulier nous allons nous concentrer sur un MERA en une dimension. Pour cela, il serait raisonnable pour un lecteur ne connaissant pas ce type d'état de s'attendre à un réseau de tenseurs ressemblant à une chaîne en 1D. Cependant, les tenseurs dans un réseau MERA parcourent à la fois la direction physique et une direction supplémentaire, qu'on appelle une direction *holographique*. Un exemple de MERA, tiré de [40], est retrouvé dans la figure 1.15.

Comme nous pouvons le voir dans la figure, un MERA est composé de deux types de tenseurs, soit des tenseurs unitaires et des tenseurs isométriques. Les isométries sont utilisées pour la normalisation entre les différentes échelles d'énergie ou de longueur, avec la plus grande taille située en haut du réseau et diminuant quand on se déplace vers le bas. Les tenseurs unitaires créent une intrication locale.

Un MERA est l'outil utilisé pour traiter les corrélations de systèmes 1D gappés ou non [16],[2]. Nous désirons observer une théorie de champ dans laquelle il existe un gap entre la première et seconde valeur propre. C'est pourquoi, dans les résultats de nos simulations, nous recherchons des corrélateurs qui diminuent de manière exponentielle avec la distance

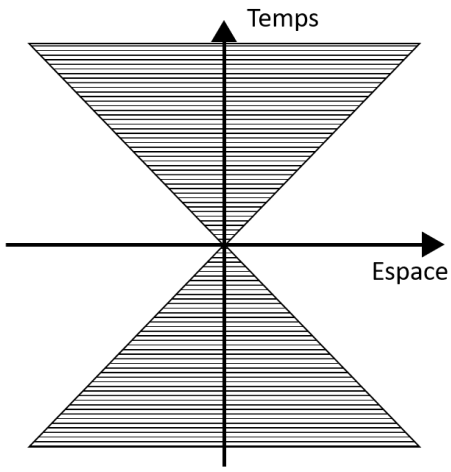


FIGURE 1.14 Une illustration standard du cône causal. L'axe horizontal est la distance spatiale du point d'origine, tandis que l'axe vertical est la distance temporelle du même point. Le cône est le parcours suivi par un photon s'éloignant de l'origine, et est représenté par la section hachurée.

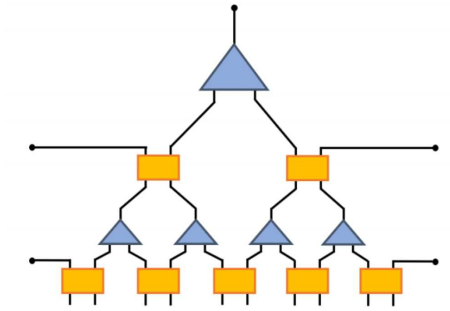


FIGURE 1.15 Un MERA. Chaque couche est composée soit d'isométries (en bleu), qui effectuent une mise à l'échelle ("coarse-graining"), ou de tenseurs unitaires (en orange), qui s'occupent de l'intrication locale.

entre deux sites. Un MERA est basé sur le concept de renormalisation de l'intrication, qui fait en sorte que les corrélations peuvent être observées sur différentes échelles de longueur. Des corrélations de différentes portées peuvent donc être étudiées sans problème sur un réseau de tenseurs qui incorpore les fondements de la structure d'un MERA. D'autant plus important, les MERAs sont connus pour être très aptes à décrire les états fondamentaux de systèmes critiques, notamment de versions discrètes de théories conformes de champs [2], en grande partie due à la capacité d'imposer l'invariance d'échelle typique des systèmes critiques.

Une autre propriété intéressante du MERA est qu'il possède un cône causal, qui limite l'effet d'une modification d'un tenseur à une partie limitée du réseau. Ce cône causal possède

une orientation différente du cône d'intrication, qui est l'ensemble des tenseurs dans la direction holographique délimités par A et $C(A)$, où $C(A)$ est la surface minimale dont la frontière correspond à la frontière de A. On illustre ces deux cônes dans la figure 1.16. C'est ici que le MERA échoue à émuler la correspondance AdS/CFT dont nous avons parlé plus haut : pour être capable de faire une reconstruction des tenseurs, il faut que le cône causal corresponde au cône d'intrication. Dans le cas d'un MERA la présence d'une orientation dans le réseau, causée par la présence d'isométries et de tenseurs unitaires, va empêcher ce phénomène de se produire [2].

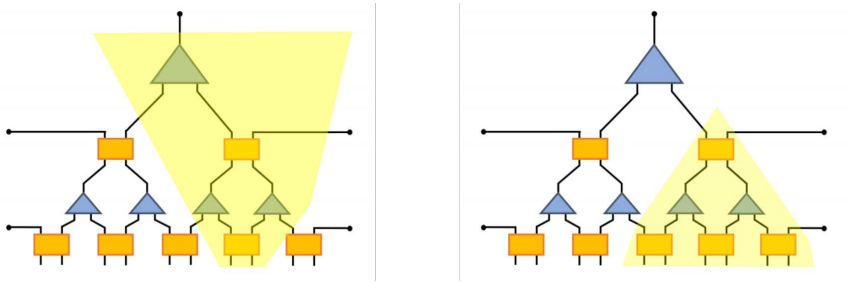


FIGURE 1.16 À gauche : Le cône causal du MERA. Un changement du tenseur unitaire en bas du réseau, ce qui revient à changer l'état physique, demande ensuite une modification de tous les tenseurs dans la section surlignée afin de correctement décrire le nouvel état. À droite : Le cône d'intrication. Une modification du tenseur en haut de la section en jaune va changer l'état des trois tenseurs oranges en bas.

Comme mentionné dans l'introduction, une solution à ce problème fût la proposition d'un code holographique composé de "tenseurs parfaits". Ces tenseurs sont construits de façon à ce que chacun d'entre eux soit une isométrie. L'uniformité du réseau enlèverait la direction favorisée du MERA, et permet en effet une reconstruction de la frontière à partir de l'information dans le volume [29]. Un astérisque important à mentionner est que cette reconstruction n'est que bien définie sur une perte de la frontière "monopartite" : une reconstruction "multipartite" apporte beaucoup de difficultés, et une reconstruction n'est pas garantie. Cependant, les fonctions de corrélation à deux points ne diminuent pas correctement avec la distance entre les deux deux points. Une théorie de champ intéressante physiquement (qui produit des états fondamentaux de systèmes critiques) présente une diminution algébrique de la corrélation entre deux sites selon la distance entre ces sites. Par contraste, les réseaux de tenseurs parfaits ne peuvent reproduire ce résultat, car ils ne possèdent que des corrélateurs indépendants de la distance entre deux sites.

Nous savons que ces réseaux appartiennent à une plus grande classe de réseaux, qui sont les réseaux de portes de parité étudiés dans ce travail. Cette classe peut produire une théorie de champ d'Ising sur la frontière du réseau en forçant la présence d'un état fermionique

sur chaque tenseur, qui est un état propre d'un hamiltonien quadratique. Des symétries additionnelles, telles que la contrainte à un seul paramètre variationnel, impose aussi l'uniformité du volume du réseau, répondant à la contrainte (1) donnée dans l'introduction. Il est aussi raisonnable de penser qu'il existe une variation du réseau, avec un certain dallage, qui pourrait représenter complètement la correspondance AdS/CFT. Le plus grand obstacle est l'absence d'un tenseur isométrique.

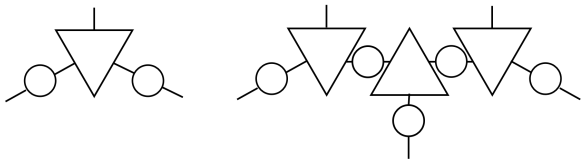


FIGURE 1.17 Deux configurations de tenseurs, chacune représentant une isométrie. Les triangles sont des tenseurs de rang 3, et les cercles des tenseurs de rang 2. Les isométries sont les tenseurs résultants de la contraction de chacune des configurations. Chaque tenseur est hyperinvariant, comme l'est décrit dans l'article d'Evenbly [2].

Afin de surmonter ce problème, Evenbly a proposé l'utilisation d'un réseau de tenseurs hyperbolique hyperinvariant [2]. Ce réseau, également uniforme et sans direction favorisée, utilise également des isométries, mais les isométries sont composées de plusieurs tenseurs dans un certain arrangement, comme l'est démontré dans la figure 1.17. Ces réseaux ne sont pas les mêmes que les réseaux de portes de parité, mais il est possible d'appliquer ce raisonnement néanmoins.

1.3 Formalisme de Grassmann et algèbre fermionique

Avant de rentrer en profondeur dans le formalisme des portes de parité, nous devons en premier introduire les outils mathématiques nécessaires pour les décrire. Étant donné que nous cherchons un état fermionique sur la frontière du réseau de tenseurs, nous pouvons utiliser le formalisme des variables anticommutantes de Grassmann, qui seront utilisées comme indices du tenseur. Nous allons définir cet ensemble, puis expliquer comment l'intégration de ces variables fonctionne, nous donnant un outil pour la contraction tensorielle. Ce formalisme sera utilisé dans les sections 1.3.1 et 1.3.2. Elles sont dérivées du travail de Bravyi sur la contraction des réseaux de portes de parité sur des graphes non-planaires [32]. Leur contenu est un résumé de celui retrouvé dans l'article.

1.3.1 Variables anticommutantes

Dans cet article, les variables de Grassmann seront dénotées par ϕ_i. Un ensemble de variables de Grassmann obéit aux mêmes règles d'anticommutation que les opérateurs de création et d'annihilation fermioniques.

$$\phi_i^2 = 0, \quad \phi_i\phi_j + \phi_j\phi_i = 0 \quad \forall i, j \tag{1.12}$$

L'algèbre $G(\phi)$ définie par l'ensemble des polynômes de $\{\phi_i, ..., \phi_N\}$ factorisés sur l'équation 1.12 possède des propriétés cruciales aux opérations que nous allons décrire dans les prochains chapitres. La base de cette algèbre est recouverte par un ensemble de monômes définis par la fonction

$$\phi(M) = \prod_{a \in M} \phi_i, \tag{1.13}$$

où $M \subseteq \{1, 2, ..., N\}$ et a augmentent en ordre croissant. Un élément f arbitraire de l'algèbre G peut donc être écrit

$$f = \sum_{M \subseteq \{1,...,N\}} f(M)\phi(M), \quad f(M) \in \mathbb{C} \tag{1.14}$$

où f et $f(\phi)$ sont interchangeables, chaque élément étant donc une fonction de ϕ_1,...,ϕ_N. Nous définissons également la fonction $I = \phi(\emptyset)$, qui est constante. Une fonction f est dénommée paire (impaire) si les monômes qui la composent ont un degré pair (impair).

1.3.2 Opérations linéaires dans l'algèbre $G(\phi)$

Partant de l'ensemble des fonctions $f \in G(\phi)$, nous allons illustrer deux fonctions linéaires essentielles au calcul des contractions sur un réseau de portes de parité ainsi qu'à l'obtention de la matrice de covariance : la dérivée et l'intégrale.

La dérivée ∂_i sur la variable ϕ_i est une application linéaire de $G(\phi)$ à $G(\phi)$. La dérivée respecte les règles d'anticommutation de l'équation 1.12. Ceci est dû à la possibilité d'exprimer la dérivée sur $f(\phi) = f_0 + \phi_i f_1$, où f_0 et f_1 sont indépendants de ϕ_i et $f_0, f_1 \in G(\phi)$, comme

$$\partial_i f = \partial f_0 + \partial(\phi_i f_1) = f_1. \tag{1.15}$$

Autrement dit, l'application $\partial_i : G(\phi) \to G(\phi)$ est définie par les règles $d_i I = 0$ et $\partial_i(\phi_j f) = \partial_i\phi_j \cdot f + \phi_i \cdot (\partial_i f)$ (règle de Leibniz). Plus tard, nous allons effectuer certains changements linéaires de variables lors de la conversion entre matrice génératrice et matrice de covariance, donc il est important de noter que ce genre de changement mène à un automorphisme de $G(\phi)$ tel que $f(\phi) \to f(\tilde{\phi})$ si l'opérateur de changement de base U est inversible. Le changement dans la dérivée pour un opérateur U est

$$\tilde{\phi}_i = \sum_{j=1}^{N} U_{i,j}\phi_j \to \tilde{\partial}_i = \sum_{i=1}^{N} U^{-1}_{i,j}\partial_j. \tag{1.16}$$

Approchons maintenant les intégrales, qui sont nécessaires pour la contraction de deux portes de parité. Une intégrale $\int d\phi_i$ sur la variable ϕ_i est une application de $G(\eta, \phi_i)$ à $G(\eta)$, où $\eta = \phi \setminus \phi_i$. Comme dans le cas des dérivées, les intégrales suivent également les règles d'anticommuation 1.12.

Prenons maintenant une intégrale sur l'ensemble ordonné $\phi = (\phi_1, ..., \phi_N)$. Cette intégrale est dénotée

$$\int D(\phi) = \int d\phi_1 ... \int d\phi_N, \tag{1.17}$$

et son action sur le monôme $\phi(M)$ est

$$\int D\phi\ \phi(M) = \begin{cases} 1 & \text{si} \quad M = \{1, 2, ..., N\} \\ 0 & \text{sinon} \end{cases}. \tag{1.18}$$

Si nous regardons cette action dans le contexte de notre définition initiale $\int D(\phi) : G(\phi, \eta) \to G(\eta)$, où η est un sous ensemble contenant les variables de Grassmann non intégrées, on retrouve :

$$\int D\phi\ \phi(M)\eta(K) = \begin{cases} \eta(K) & \text{si} \quad M = \{1, 2, ..., N\} \\ 0 & \text{sinon} \end{cases}. \tag{1.19}$$

Les changements de variables demeurant tout aussi importants que pour la dérivée, observons l'effet d'un changement de variables $\tilde{\phi}_i = \sum_{j=1}^{N} U_{i,j}\phi_j$ sur l'intégrale $\int D(\phi)$. La nouvelle intégrale devient

$$\int D\tilde{\phi} = \det(U) \int D\phi. \tag{1.20}$$

Pourquoi est-ce intéressant ? La majorité des transformations que nous allons effectuer sont des opérateurs unitaires dans l'espace gaussien. Ces opérateurs ont un déterminant de 1, ce qui évitera l'apparition de termes supplémentaires devant l'intégrale.

1.3.3 Équivalence entre les bases

Bien que nous avons introduit d'abord les variables de Grassmann, qui seront dorénavant interchangeables avec les opérateurs de création et d'annihilation fermioniques, nous rappelons que le système que nous désirons avant tout étudier est effectivement un système de spins. L'interchangeabilité des variables est due au fait que les deux respectent les équations 1.12. Nous allons utiliser les notions introduites jusqu'ici dans nos calculs, mais afin de les représenter d'une manière efficace et intuitive dans ce mémoire, nous allons effectuer une transformation de Jordan-Wigner afin de convertir N spins en $2N$ modes de Majorana γ. Les modes de Majorana sont définis comme

$$\gamma_{2i-1} = Z_1 Z_2 ... Z_{i-1} X_i \tag{1.21}$$
$$\gamma_{2i} = Z_1 Z_2 ... Z_{i-1} Y_i \tag{1.22}$$

où les opérateurs de spins sur le site i sont définis en termes des opérateurs σ^x, σ^y et σ^z par

$$X_i = I_2^{\otimes(i-1)} \otimes \sigma^x \otimes I_2^{\otimes(N-i)} \tag{1.23}$$
$$Y_i = I_2^{\otimes(i-1)} \otimes \sigma^y \otimes I_2^{\otimes(N-i)} \tag{1.24}$$
$$Z_i = I_2^{\otimes(i-1)} \otimes \sigma^z \otimes I_2^{\otimes(N-i)} \tag{1.25}$$

Nous allons également définir l'opérateur de parité global (P_{tot}) par

$$P_{tot} = Z_1 Z_2 ... Z_N = (-i)^N \gamma_1 \gamma_2 ... \gamma_{2N}. \tag{1.26}$$

C'est avec cet opérateur que nous allons définir la parité d'une porte de parité, avec la valeur propre de P_{tot} égale à 1 pour un tenseur pair et égale à -1 pour un tenseur impair.

La relation entre les variables de Grassmann et les modes de Majorana est la suivante [44] :

$$f_i = \frac{1}{2}(\gamma_{2i} + i\gamma_{2i-1}) \tag{1.27}$$

$$f_i^\dagger = \frac{1}{2}(\gamma_{2i} - i\gamma_{2i-1}) \tag{1.28}$$

car les modes de Majorana ont la propriété $\gamma_i = \gamma_i^\dagger$. On peut donc faire une équivalence entre la configuration fermionique d'un système et sa configuration en paires de Majorana. Ces paires de modes adjacents sont dénommées "dimères" [45]. Un dimère est une paire de modes fermioniques dans un état singulet. À l'origine les dimères furent utilisés pour résoudre des systèmes de spins en deux dimensions, comme les liaisons de valence résonantes dans des aimants antiférromagnétiques ou des liquides de spin Z_2 ([46], [47]), mais plus récemment certains recherchent leurs avantages dans la correction d'erreur avec les codes stabilisateurs ([29],[44],[48]).

Un hamiltonien quadratique

$$H = \frac{i}{2}\sum_{i,j\in\Omega} p_{i,j}\gamma_i\gamma_j \tag{1.29}$$

où $\Omega = \{(i,j)\}$ possède un état fondamental $|\psi_f\rangle$ contenant N états fermioniques qui est annulé par N équations avec la forme

$$(\gamma_i + ip_{i,j}\gamma_j)\,|\psi_f\rangle = 0. \tag{1.30}$$

Chacune de ces équations correspond à un dimère. La preuve de cette démarche est retrouvée dans le deuxième article de Jahn *et al.* sur ce sujet [44]. Ces dimères ont une visualisation graphique très intuitive, illustrée dans la figure 1.18. La flèche est orientée dans la direction $i \to j$, et la couleur de la flèche indique la parité de $p_{i,j}$, avec noir pour $p_{i,j} = 1$ et rouge pour $p_{i,j} = -1$. Les valeurs possibles de $p_{i,j}$ sont situées dans l'intervalle [-1,1], donc la couleur de la flèche ne dépendra que du signe de $p_{i,j}$. Nous avons remplacé dans cette figure les sites fermioniques par les sites des fermions de Majorana. Chaque paire de site sur la même arête correspond au site fermionique représenté par ces deux modes de Majorana.

À partir de cette visualisation, il est aisé de faire la transition vers la matrice de covariance, de laquelle nous pouvons calculer les observables voulues du système et observer leur diminution algébrique avec l'augmentation de la distance entre deux sites. La matrice de

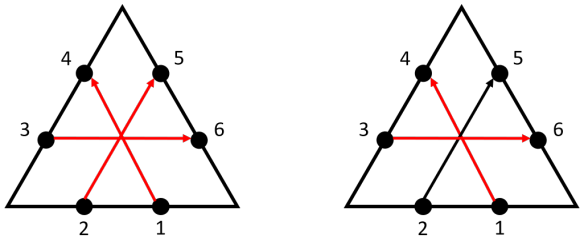


FIGURE 1.18 Deux dimèrisations. Les flèches représentent des entrées non-nulles dans la matrice de covariance, avec -1 pour une flèche rouge et +1 pour une flèche noire. Le tenseur à gauche est le $|0\rangle$ logique d'un dallage {3,7} et le tenseur à droite est le $|1\rangle$ logique du même dallage.

covariance est de forme

$$\Gamma_{i,j}(\psi) = \langle\psi| \frac{i}{2}[\gamma_i, \gamma_j] |\psi\rangle . \tag{1.31}$$

Les entrées de cette matrice sont dans l'intervalle [-1,1] selon la valeur de $p_{i,j}$ pour γ_i et γ_j. Par exemple, la matrice de $|\tilde{0}\rangle$ est

$$\Gamma(\tilde{0}) = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \end{pmatrix} \tag{1.32}$$

Une autre perspective sur les dimères, qui sera l'introduction sur le formalisme des porte de parité, est une interprétation graphique. La combinaison de tous les dimères crée une carte d'appariement, aussi appelée "matching", sur le système complet. De cette carte, il est possible de calculer un Pfaffian, qui permet de retrouver la fonction caractéristique qui définit une porte de parité, qui sera définie entièrement dans la section suivante.

1.4 Portes de parité

Nous avons donné une définition très générale des portes de parité lors de l'introduction. Rentrons maintenant plus en détails dans une définition précise des portes de parité. Les portes de parité ont été créées par Leslie Valiant en 2002 [31] dans le contexte de l'information quantique en tant que manière d'efficacement simuler des circuits quantiques avec un ordinateur classique. Son travail a été amélioré par Knill, Terhal, Divincenzo, Bravyi et Jozsa, qui ont à la fois adapté et étendu le formalisme des portes de parité aux fermions et aux portes à 2 qubits ([32], [49]-[50]).

1.4.1 Définition graphique

Prenons un tenseur T de rang r avec entrées $x \in \{0,1\}^{\otimes r}$. T est une porte de parité si, pour une matrice antisymétrique $A \in \mathbf{C}^{r\,x\,r}$ et $z \in \{0,1\}^{\otimes r}$,

$$T(x) = \mathrm{Pf}(A_{|x\ \mathrm{XOR}\ z})T(z). \tag{1.33}$$

$T(x)$ est l'élément de T désigné par l'indice x. $\mathrm{Pf}(A_{|x})$ ici est le Pfaffian de la sous-matrice de A désignée par x. Qu'est-ce qu'un Pfaffian ? Nous le définissons simplement comme une valeur intrinsèque d'une matrice antisymétrique, au même niveau que le déterminant, mais cela ne satisfait pas les besoins de cette section. Pour répondre à cette question en détails, nous retournons aux dimères et à leur représentation graphique, qui rend le calcul normalement difficile du Pfaffian plus accessible. Nous voyons dans la figure 1.19 que les dimères recouvrent entièrement le réseau de sites, pairant les sites adjacents. Par ceci ils créent ce qu'on appelle un "agencement parfait". C'est par cet agencement que nous allons retrouver le Pfaffian.

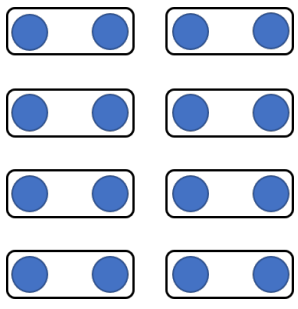


FIGURE 1.19 Un recouvrement d'un système de spins. Chaque point bleu représente un site contenant un spin. Les carrés sont les combinaisons de deux sites recouverts par un dimère.

Afin de bien décrire ce qu'est un agencement parfait, commençons par un rappel de la notation d'un graphe G. Un graphe $G = (V, E, W)$ désigne un graphe pondéré avec un ensemble de sommets V, un ensemble d'arêtes E et une fonction W qui donne une pondération $W(e)$ à chaque arête $e \in E$. Il est également possible de définir un élément e comme une paire non-ordonnée (i, j) des sommets $i, j \in V$. Un exemple de ce genre de graphe est présent dans la figure 1.20.

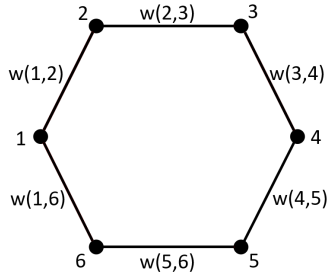


FIGURE 1.20 Un graphe pondéré non-orienté. Chaque point noir est un sommet, et les lignes noires sont des arêtes. Le numéro adjacent à chaque ligne est le poids associé avec l'arête.

Partant de cette notation, un agencement est un sous-ensemble des arêtes $M \subseteq E$ dans lequel chaque sommet $v \in V$ a au plus une arête incidente de M. Deux définitions sont bâties sur celle-ci. Un agencement imparfait est un agencement dans lequel un sous-ensemble $S \subseteq V$ ne possède aucune arrête incidente provenant de M. Ces agencements sont nommés S-imparfaits. Un agencement parfait est un agencement pour lequel $S = \emptyset$. Ces deux agencements sont illustrés dans la figure 1.20. Nous pouvons voir que l'agencement dans la figure 1.21 détient deux sommets qui ne sont pas connectés par une arête dans M. Il est évident par définition qu'un graphe avec un nombre impair de sommets ne contient pas d'agencement parfait, car un sommet sera toujours présent dans S.

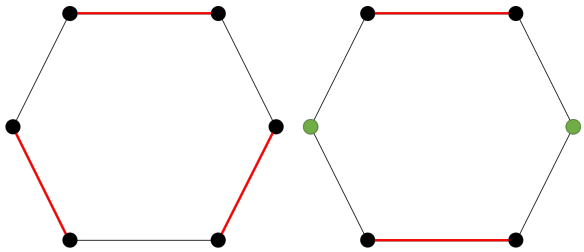


FIGURE 1.21 Deux graphiques, illustrant un agencement parfait (à gauche) et un agencement imparfait (à droite). Les arêtes surlignées en rouge représentent les arêtes de l'agencement. Les deux sommets en verts dans l'agencement imparfait sont les sommets exclus de l'agencement.

L'ensemble des agencements S-imparfaits pour un graphe G donné est définit $K(G, S)$.

D'ici, nous définissons la somme d'agencement

$$I(G,S) = \sum_{M \in K(G,S)} \prod_{e \in M} W(e). \tag{1.34}$$

L'étape clé est ensuite de démontrer la relation entre cette somme et le Pfaffian. Heureusement, un algorithme pour cela a déjà été introduit par [51]. Le théorème qui s'ensuit est

Théorème 1. Pour un graphe pondéré $G = (V, E, W)$ avec $O(n^2)$ sommets et $O(n^2)$ arêtes il existe une matrice antisymétrique A de taille $n \times n$ pour laquelle les coefficients $A_{i,j}$ peuvent être calculés à partir des poids $W(e)$, et dont le Pfaffian peut être relié à la somme d'agencement par

$$\mathrm{Pf}(A) = \sum_{M \in K(G,\emptyset)} \prod_{e \in M} W(e), \tag{1.35}$$

avec l'algorithme Fisher-Kasteleyn-Temperley. Le Pfaffian de A peut être calculé en temps polynomial $O(n^2)$.

La preuve complète de ce théorème est retrouvée dans [32], mais nous allons donner une démarche intuitive ici. Le Pfaffian d'une matrice antisymétrique n x n est définie comme étant 0 si n est impair, et

$$\mathrm{Pf}(A) = \sum_{\sigma} \epsilon_\sigma w(\sigma(1), \sigma(2)) w(\sigma(2), \sigma(3)) \cdots w(\sigma(n-1), \sigma(n)) \tag{1.36}$$

si n est pair [31]. On somme sur toutes les permutations σ de $[1, 2, 3, ..., n]$, et $\epsilon_\sigma \in \{-1, 1\}$ est le signe de la permutation de σ. Ce signe représente la parité du nombre de transpositions requises pour ordonner σ dans la permutation identité. Ces transpositions sont requises dû à la nature planaire du graphe. De la forme 1.36, une correspondance intuitive se forme entre le Pfaffian et l'agencement d'un graphique.

Nous avons donc maintenant une manière de calculer le Pfaffian, en le mettant équivalent avec la somme d'agencement. Donnons ensuite un lien entre l'agencement et un élément du tenseur $T(x)$. Pour ce faire nous exprimons un tenseur de rang n par

$$T = \sum_{x \in \{0,1\}^{\otimes n}} T(x) f_1^{\dagger x_1} f_2^{\dagger x_2} \cdots f_n^{\dagger x_n} |\emptyset\rangle . \tag{1.37}$$

Afin de simplifier l'exemple, prenons un tenseur T pair avec $n = 3$ représentant l'état de trois fermions. Ces deux contraintes imposent $T(x) = 1, \forall x \in \{0,1\}^{\otimes 3}$ tel que $\sum_i x_i = 2$

ainsi que pour $x = 0^{\otimes 3}$, et $T(x) = 0$ autrement. Illustrons maintenant ce modèle dans la figure 1.22. Chaque sommet représente maintenant un mode fermionique sur un des sites. Si deux sites possèdent un fermion, nous allons ajouter une arrête entre les deux sites. Cette arête aura un poids associé à la valeur de l'élément du tenseur correspondant à l'état. Par exemple, regardons un tenseur T_ψ définit par l'état

$$|\psi\rangle = T(000)\,|000\rangle + T(110)\,|110\rangle + T(101)\,|101\rangle + T(011)\,|011\rangle\,, \tag{1.38}$$

où $T(x)$ sont les coefficients de cet état après la normalisation. Les trois sites dans la figure 1.22 étant identifiés, on peut voir que le poids de l'arête correspond à $T(011)$ dans 1.22a. On peut également remarquer que ce graphique est un agencement 1-imparfait, donc avec le sommet 1 exclu. Regardant 1.22b et 1.22c, on remarque que les agencements sont 2-imparfait et 3-imparfait, respectivement. Nous concluons donc que les poids associés sont égaux à $T(101)$ (2-impafait), et $T(110)$ (3-imparfait). Bâtir le graphe illustré revient dans ce cas à construire la fonction d'état représentée par le tenseur $T(\phi)$. Une question demeure, qui est bien sûr comment retrouver $T(000)$. La réponse est de sortir le coefficient $T(000)$, que nous allons redéfinir Z^{-1}, et de retrouver des nouveaux coefficients $T'(x)$ donnant l'état

$$|\psi\rangle = Z^{-1}(|000\rangle + T'(110)\,|110\rangle + T'(101)\,|101\rangle + T'(011)\,|011\rangle), \tag{1.39}$$

où $T'(x) = Z * T(x)$. Les poids du graphe G seront changés à $T'(x)$, et la valeur de Z sera facile à retrouver en calculant $\langle\psi|\psi\rangle$.

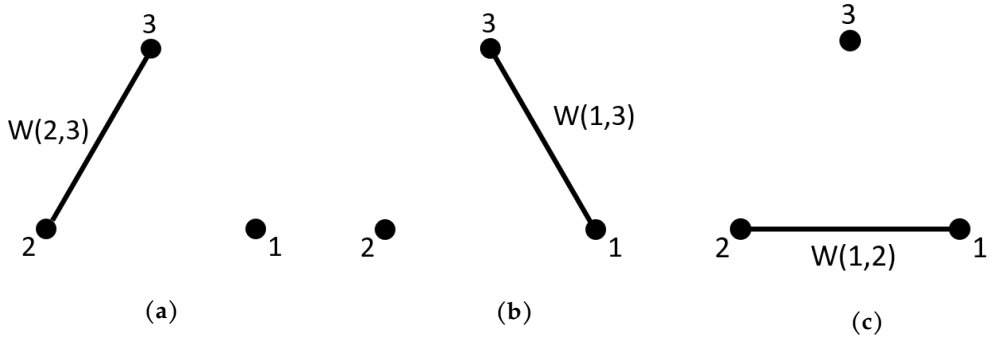


FIGURE 1.22 Trois agencements imparfaits. (a) est 1-imparfait, (b) est 2-imparfait et (c) est 3-imparfait.

L'équivalence que nous avons démontrée à main levée ici est résumée par l'équation

$$T(x) = \sum_{M \in K(G,\{i|\forall x_i=0\})} \prod_{e \in M} W(e), \tag{1.40}$$

où $W(e)$ relie deux sites occupés. De cette équivalence et de la généralisation de l'équation 1.35

$$\mathrm{Pf}(A_{|S}) = \sum_{M \in K(G,S)} \prod_{e \in M} W(e), \tag{1.41}$$

dans laquelle $A_{|S}$ est le Pfaffian de la sous-matrice principale agissant sur le sous-espace supporté par S, on retrouve le théorème principal de cette section.

Théorème 2. Pour un graphe pondéré $G = (V, E, W)$ avec $O(n^2)$ sommets et $O(n^2)$ arêtes et un sous-ensemble de sommets $V_{ext} \subseteq V$, il existe un tenseur associé $T(\psi)$ pour lequel chacun des coefficients $T(x)$ où $x \subseteq V_{ext}$ peut être calculé dans un temps $O(n^2)$ par

$$T(x) = \mathrm{Pf}(A_{|x}), \tag{1.42}$$

où A est une matrice antisymétrique telle que les coefficients de A peuvent être calculés à partir de $W(e)$.

L'isotropie de T va forcer une symétrie dans A, soit que le Pfaffian doit demeurer invariant sous permutation cyclique des indices de A. Pour que cela soit vrai, nous devons limiter A à un degré de liberté, donc les coefficients de la matrice seront tous égaux. Nous nommons ce paramètre libre a. La matrice A aura la forme

$$A = \begin{pmatrix} 0 & a & a \\ -a & 0 & a \\ -a & -a & 0 \end{pmatrix}. \tag{1.43}$$

À ce point, il est important de mentionner deux points. Premièrement, une telle méthode ne fonctionne que pour un état pair. En effet, comme nous allons le mentionner dans la section suivante, la formule nécessaire à l'obtention d'un état impair demande certains termes supplémentaires. Finalement, les éléments de la matrice A sont dénommés *corrélateurs* dans la littérature, et nous allons suivre cette convention dans le mémoire. Il sera nécessaire de différencier ces corrélateurs avec ceux obtenus à partir de la matrice de covariance Γ. Une manière de distinguer les deux est de garder en tête qu'un corrélateur dans la matrice A ne donne que la relation entre deux opérateurs de création fermioniques, tandis que le corrélateur obtenu de Γ représente une quantité physique calculée à partir d'une observable.

1.4.2 Équivalence avec fermions libres

Partant du résultat obtenu dans la dernière section, nous allons maintenant démontrer l'équivalence entre les portes de parité et un état pair de fermions libres en utilisant le Pfaffian. Pour ce faire, nous allons d'abord définir une fonction génératrice pour le Pfaffian, dont la démonstration se retrouve dans [32]. La fonction génératrice est

$$\mathrm{Pf}(A) = \int D\theta \exp\left(\frac{1}{2}\theta^T A\theta\right). \tag{1.44}$$

Nous utilisons le même formalisme que celui introduit dans la section 2.2. L'exponentielle est la clé pour lier le Pfaffian à une théorie de fermions libres. Commençons en prenant un hamiltonien gaussien H_G

$$H_G = \frac{i}{4}\sum_{m,n=1}^{2N} G_{m,n}\gamma_m\gamma_n. \tag{1.45}$$

Cet hamiltonien a des vecteurs propres $|\psi_G\rangle$ de la forme $U_G|\emptyset\rangle$, avec U_G un opérateur unitaire gaussien. Nous allons définir $|\emptyset\rangle$ comme l'état du vide. La matrice G est antisymétrique et réelle, et les γ sont les opérateurs auto-adjoints de Majorana. Partant des définitions données dans la section 2.2, on retrouve leurs relations, qui sont les relations de Clifford soit $\{\gamma_m, \gamma_n\} = 2\delta_{m,n}I_2$.

Partant d'une réduction Bloch-Messiah [52], il existe une transformation unitaire qui permet de découpler le vecteur d'état $|\psi_G\rangle$. Nous désirons que la transformation préserve l'intégrité du vide $|\emptyset\rangle$, ce qui conserve le nombre de particules. Cette transformation met en forme normale à la fois les relations de saut et les termes de couplage, $f_i^\dagger f_j$ et $f_i f_j$ respectivement. Pour cela, nous utilisons la méthode développée par Bogoliubov, qui mettra les termes de saut en forme diagonale et les termes de couplage en forme bloc-diagonale. Nommons l'opérateur qui effectue cette transformation $U_B \in \mathbb{C}^{N\mathrm{x}N}$, qui sera défini par la relation

$$\tilde{f}_m^\dagger = U_B f_m^\dagger U_B^\dagger = \sum_{m,m'} U_{m,m'} f_{m'}^\dagger. \tag{1.46}$$

Le vecteur d'état $|\psi_G\rangle$ peut donc être exprimé

$$|\psi_G\rangle = U_B \prod_{m=1}^{N/2} (v_m + u_m f_{2m-1}^\dagger f_{2m}^\dagger)\,|\varnothing\rangle = \prod_{m=1}^{N/2} (v_{2m-1} I_2 + u_{2m} \tilde{f}_{2m-1}^\dagger \tilde{f}_{2m}^\dagger)\,|\varnothing\rangle\,. \tag{1.47}$$

Prenant $v_m \neq 0$ et $\tilde{f}_m^{\dagger 2} = 0$, on peut exprimer ce produit en terme d'exponentielle

$$|\psi_G\rangle = \prod_{m=1}^{N/2} v_{2m-1} \exp\left(\frac{u_{2m}}{v_{2m-1}} \tilde{f}_{2m-1}^\dagger \tilde{f}_{2m}^\dagger\right)|\varnothing\rangle \tag{1.48}$$

$$= \sqrt{Z}^{-1} \exp\left(\sum_{m=1}^{N/2} \frac{u_{2m}}{2v_{2m-1}} [\tilde{f}_{2m-1}^\dagger, \tilde{f}_{2m}^\dagger]\right)|\varnothing\rangle\,, \tag{1.49}$$

avec $\sqrt{Z}^{-1} = \prod_{m=1}^{N/2} v_{2m-1}$ et utilisant les relations d'anticommutation pour calculer le commutateur. Regardons cet anticommutateur de plus près : son existence montre clairement que l'exponentielle peut être exprimée avec une matrice hermitienne diagonale par bloc. Définissons $\lambda_m = u_{2m}/(2v_{2m-1})$, cette matrice L est définie par l'équation

$$L = \bigoplus_{m=1}^{N} \begin{pmatrix} 0 & \lambda_m \\ -\lambda_m & 0 \end{pmatrix} \tag{1.50}$$

si N est pair, et ajoutant un bloc 0 x 0 si N est impair. Nous utilisons ensuite le fait que toute matrice antisymétrique peut être exprimée dans une forme normale dénotée par A_N par une transformation unitaire afin d'écrire l'équation 1.47 comme

$$|\psi_G\rangle = \sqrt{Z}^{-1} \exp\left(\sum_{m,n=1}^{N} A_{m,n} f_m^\dagger f_n^\dagger\right)|\varnothing\rangle\,, \tag{1.51}$$

où $A = U A_N U^\dagger$. On voit que cette définition ramène un terme présent dans la fonction génératrice du Pfaffian (1.44). En effet, dérivant cette équation, on retrouve la relation

$$\sum_{x \in \{0,1\}^{\otimes N}} \mathrm{Pf}(A_{|x})\theta^{x_1}\theta^{x_2}\cdots\theta^{x_N} = \exp\left(\sum_{m,n=1}^{N} A_{m,n}\theta_m^\dagger \theta_n^\dagger\right). \tag{1.52}$$

Remplaçant les opérateurs gaussiens pour les opérateurs de création, on retrouve la formule correspondant à l'équation 1.51. On voit donc que pour le tenseur T_G représentant l'état gaussien $|\psi_G\rangle$, l'élément $T_G(x)$ est donné par $\sqrt{Z}^{-1}\mathrm{Pf}(A_{|x})$, donc le tenseur est une porte

de parité.

1.4.3 Portes à deux qubits

Certains articles ont déjà démontré l'utilisation possible des portes de parité en informatique quantique ([31], [50]). Ici nous allons expliquer l'application de portes de parité de rang 4 faite par Josza en 2008, dans laquelle chaque porte de parité devient une porte à 2 qubits dans un circuit. Non seulement est-ce une introduction supplémentaire aux portes de parité, mais cet exemple nous permettra d'illustrer le point clé initial derrière la création des portes de parité par Valiant, soit que les circuits quantiques de portes de parité sont simulables classiquement en temps polynomial.

Commençons par l'introduction des portes logiques à deux qubits. Un exemple est situé dans la figure ci-dessous.

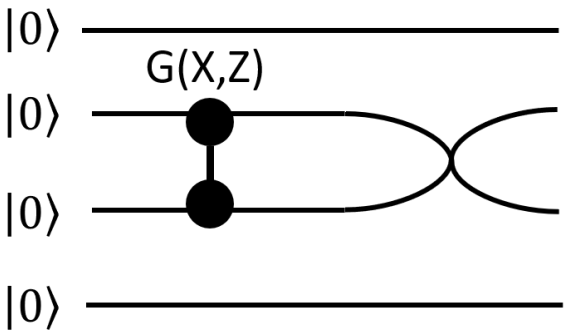


FIGURE 1.23 Une porte de parité affectant le deuxième et troisième qubit. Le croisement suivant la porte G(X,Z) est la porte logique SWAP.

Cette porte est la porte $G(X,Z)$, qui applique un opérateur X sur les états pairs et un opérateur Z sur les états impairs. La matrice qui représente cette porte est

$$G(X,Z) = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \tag{1.53}$$

qui peut être décomposée en la matrice X et Z de la manière présentée dans la figure 1.24, où $A = X$ et $B = Z$.

$$G(A,B) = \begin{pmatrix} p & 0 & 0 & q \\ 0 & w & x & 0 \\ 0 & y & z & 0 \\ r & 0 & 0 & s \end{pmatrix} \qquad A = \begin{pmatrix} p & q \\ r & s \end{pmatrix} \quad B = \begin{pmatrix} w & x \\ y & z \end{pmatrix}$$

FIGURE 1.24 La décomposition d'une porte de deux qubits en deux portes, avec les éléments dans l'encadrement en vert se retrouvant dans la matrice A et ceux dans celui en rouge se retrouvant dans la matrice B.

Dans l'article de Jozsa, deux contraintes sont introduites sur les matrices A et B. (1) Elles doivent être dans U(2) ou SU(2), et (2) elles doivent posséder le même déterminant. L'action de A est restreinte au sous-espace pair, soit $|00\rangle$ et $|11\rangle$, et B agit sur le sous-espace impair $|10\rangle$ et $|01\rangle$. À partir de ces portes, nous citons un théorème obtenu par Jozsa [50].

Théorème 3. Considérons une famille de circuits C_n composés de portes $G(A,B)$ respectant les contraintes suivantes :

- $G(A,B)$ sont appliquées sur les indices voisins seulement.

- L'état initial est un état produit.

- La mesure est effectuée par un opérateur Z sur le premier qubit.

Cette famille de circuit peut être simulée de manière classique avec une augmentation constante dans la taille du circuit, c'est-à-dire que les probabilités de la mesure peuvent être calculées en temps polynomial en fonction du nombre de qubits (n) et du nombre de bits de précision désiré.

Ce groupe peut être agrandi pour former un groupe universel, cependant, en introduisant la porte *SWAP*. Cette porte échange deux indices, et sa matrice est

$$SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{1.54}$$

Il est possible de voir que cette matrice ne respecte pas les restrictions imposées sur $G(A,B)$ en la décomposant en

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad B = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \tag{1.55}$$

avec det$(A) = 1$ et det$(B) =$ -1. Ce résultat n'est pas conforme à la condition (2) des portes de parité, et donc on peut conclure que SWAP n'en est pas une. Pourquoi SWAP permettrait au groupe de portes d'être complet ? Pour répondre à cela, retournons au théorème 3. Les deux premières contraintes, soit délimiter la distance d'interaction entre les qubits au voisin le plus proche et l'absence d'intrication dans l'état initial, semblent restreindre l'impact de l'action dans le circuit, d'où le concept d'action limitée. La porte SWAP enlève cette limite. Regardons la figure 1.25. L'action de $G(A, B)$ n'affecte que les deux premiers qubits, mais en utilisant la porte SWAP trois fois nous voyons que le circuit revient à l'application de la porte $G(A, B)$ sur les qubits 1 et 5. Dans ce cas, les contraintes du théorème 3 ne sont plus respectées, et le circuit ne peut pas nécessairement être simulé efficacement utilisant des méthodes classiques.

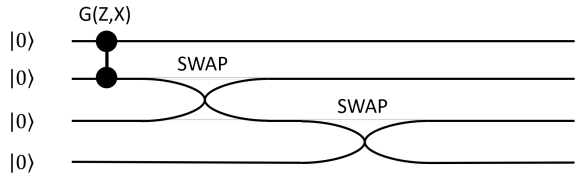


FIGURE 1.25 Une application de la porte SWAP pour étendre l'influence des portes à 2 qubits. La porte n'affecte originalement que les deux premiers qubits. Le premier SWAP change cet impact sur le premier et le troisième qubit, et la deuxième porte SWAP étend cette influence sur le premier et le quatrième qubit.

Bien que cela veut dire qu'on ne peut obtenir un ensemble de portes universel, un certain aspect positif ressort de cette limitation. Un opérateur causant une erreur sur un qubit dans le code aura son impact restreint, permettant une certaine correction d'erreur si assez peu de qubits sont affectés par cette erreur. Présentement les seules erreurs étudiées dans ce genre de code sont les erreurs d'effacement [29] et les erreurs de Pauli [53]. On peut comparer cet effet à celui retrouvé dans un code holographique, dans lequel une perte de certains qubits à la surface avait un impact limité sur la profondeur de la perte d'information.

Revenons maintenant sur le temps polynomial de simulation d'un circuit, utilisant des éléments que nous avons développés dans les sections précédentes. Comme pour les opérateurs de Majorana, on définit $2n$ opérateurs $\{c\}$ qui obéissent aux relations d'anticommutation $\{c_\mu, c_\nu\} = 2\delta_{\mu,\nu} I$, avec $\mu, \nu = 1, 2, ..., 2n$. Ces relations définissent une *algèbre de Clifford* C_{2n}, qui est une algèbre définie sur un espace vectoriel muni d'une forme quadratique. Ces $2n$ opérateurs agissent sur chacun des n qubits comme un opérateur fermionique

de position ($\mu = 2k-1$, où $k \in \{1,...,n\}$) ou momentum ($\mu = 2k$). Chaque élément de l'algèbre peut être exprimé

$$\sum_{i_1<...<i_k} A_{i_1,...,i_k} c_{i_1} \cdots c_{i_k}. \tag{1.56}$$

L'espace de Clifford est donc un espace vectoriel de dimension $2^{2n} = 2^n$ x 2^n, et peut être représenté par une matrice de taille n x n. Ceci est la taille de la matrice génératrice antisymétrique que nous avons définie dans la section précédente. C'est grâce à cette propriété que nous allons pouvoir représenter des états de taille $n \geq 300$, ce qui serait autrement impossible si nous devions garder en mémoire 2^{300} coefficients.

Prenons un élément de C_{2n} de forme quadratique $H = i\sum_{\mu,\nu} h_{\mu\nu} c_\mu c_\nu$, où $h_{\mu\nu}$ est une matrice antisymétrique. Comme dans le calcul d'équivalence entre porte de parité et fermion, on trouve un opérateur gaussien unitaire définit $U_c = e^{iH}$ tel que

$$U_c^\dagger c_\nu U_c = \sum_\mu R_{\nu\mu} c_\mu. \tag{1.57}$$

R est dans SO(2), ce qui veut dire que l'expression $U_c^\dagger c_\nu U_c$ demeure dans un sous-espace de dimension $2n$. Nous avons donc besoin de ne garder en mémoire que les générateurs de l'espace. Étant donner que les portes $G(A,B)$ sont des portes de parité, toute action sur les qubits revient à l'application d'un opérateur gaussien, ce qui mène au prochain théorème de Jozsa [50].

Théorème 4. Pour un circuit de taille polynomiale composé de portes gaussiennes agissant sur un état produit $|\psi\rangle_{in}$, un observable Z_k aura une valeur correspondante $\langle Z_k\rangle_{out}$ de taille $O(n^d)$ si l'observable peut être exprimé dans C_{2n} comme un polynôme de degré d. Donc, $\langle Z_k\rangle_{out}$ sera calculable en temps polynomial si d n'augmente pas avec n. [50]

$$|\psi\rangle_{out} = U_c |\psi\rangle_{in} \rightarrow \tag{1.58}$$

$$\langle c_\mu\rangle_{out} = \langle\psi| U_c^\dagger c_\mu U_c |\psi\rangle_{in} = \sum_\nu R^c_{\mu\nu} \langle\psi| c_\nu |\psi\rangle_{in}, \tag{1.59}$$

où $|\psi\rangle_{in}$ est l'état d'entrée et $|\psi\rangle_{out}$ est l'état final. Sachant que $Z_k = -ic_{2k-1}c_{2k}$ (les mêmes relations que les opérateurs de Majorana), on retrouve la preuve aisément.

$$\begin{aligned}\langle Z_k\rangle_{out} &= \langle -ic_{2k-1}c_{2k}\rangle_{out} = -i\,\langle\psi|\,U_c^\dagger c_{2k-1}c_{2k}U_c\,|\psi\rangle_{in} \\ &= -i\,\langle\psi|\,U_c^\dagger c_{2k-1}UU^\dagger c_{2k}U_c\,|\psi\rangle_{in} = \sum_{\mu\nu} R^c_{2k-1\mu}R^c_{2k\nu}\,\langle\psi|\,c_\mu c_\nu\,|\psi\rangle_{in}\end{aligned}$$

Chapitre 2

Algorithme

La fondation théorique maintenant établie, nous allons élaborer sur l'implémentation du code permettant de manipuler un tel réseau. Le code pour le calcul de la matrice de covariance, qui est celle utilisée dans le calcul des corrélateurs, a été fait dans MATLAB. Afin de faciliter la compréhension et présenter des exemples uniformes à travers le code, les exemples suivants utiliseront un dallage {5,4} ou {3,7}, et les indices seront de dimension 2. Ce code est presque universel, permettant de créer un réseau de tenseurs hyperbolique pour n'importe quel dallage hyperbolique ou planaire, et accommodant des dimensions d'indices supérieures à 2. Cependant les opérations sur les tenseurs eux-mêmes exigent que ceux-ci soient pairs. La fonction caractéristique des tenseurs impairs possède une forme différente, soit une intégration sur une variable de Grassmann auxiliaire. Ils ne seront pas inclus ici, car leur traitement est significativement plus compliqué et n'était pas nécessaire à la réalisation de ce projet.

À l'origine le code était écrit dans le langage de programmation Julia, qui est normalement très efficace dans le traitement de réseaux de tenseurs, étant le langage de choix pour des programmes tels iTensor ou TensorFlow. Cependant, dû à la nature particulière des portes de parité, MATLAB est un langage beaucoup plus compatible avec notre code. En effet, pour le calcul d'un réseau de tenseurs avec 324 sites sur la frontière, MATLAB est capable de retourner la matrice de covariance correspondante en moins d'une minute, tandis que Julia peut prendre jusqu'à 10 minutes pour faire le même calcul. Cela dit, il est important de remarquer que ces tests ont été faits avant que la version finale du code ait été programmée, et donc ces temps peuvent changer si le code était reproduit dans Julia.

Chaque section du chapitre introduira une étape dans le calcul des corrélateurs d'un réseau hyperbolique. Le code sera inclus en sa totalité, accompagné d'explications écrites et

visuelles. La première section illustre la manière avec laquelle la structure et le dallage du réseau sont créés en mémoire. Cette section inclus trois méthodes différentes, dont deux ayant déjà été codées. La troisième méthode ne sera que discutée en grandes lignes. La section 2.2 contient le code utilisé dans la contraction et manipulation du réseau de tenseurs, incluant les trois opérations fondamentales : la permutation cyclique, la contraction et l'auto-contraction. La section suivante explique la démarche requise pour la transition entre matrice génératrice et matrice de covariance, qui permettra par la suite de calculer l'intrication et les autres corrélateurs, ce qui sera élaboré dans la section 2.4.

2.1 Création du réseau

Ceci est la première étape de l'algorithme générant la matrice de covariance finale. La création du réseau dont nous parlons ici n'est pas le calcul du réseau en soi, mais plutôt la formation d'une liste d'adjacence indiquant les positions et voisins de chaque tenseur composant le réseau. Dans cette section, nous proposons trois méthodes différentes pour le calcul de réseau : la méthode de croissance *Blossom* (section 2.1.1), la méthode de croissance *concentrique* (section 2.1.2) et la méthode de croissance *géodésique* (section 2.1.3). Seules les deux premières méthodes seront présentées avec leur code correspondant, l'implémentation de la troisième méthode possédant un degré de difficulté vastement supérieur sans présenter un avantage particulièrement intéressant dans le contexte de ce projet. Nous allons commencer en présentant ces algorithmes, pour ensuite illustrer leurs différences principales et indiquer quel algorithme est optimal selon divers ensembles de critères.

2.1.1 Algorithme Blossom

Avant de rentrer dans les détails du code, une visualisation de l'algorithme est donnée par la figure 2.1. Le principe général de l'algorithme est de débuter avec un tenseur solitaire, puis d'attacher un tenseur à chaque indice libre. Ce processus est répété le nombre de fois voulu, résultant en un réseau de tenseurs qui possède une distance absolue uniforme entre n'importe quel tenseur sur sa frontière et le tenseur central. Dans ce contexte, la distance absolue est définie comme étant le nombre de tenseurs connectant un tenseur au tenseur central inclusivement. La figure 2.2 montre ici un réseau avec une distance de 5.

L'algorithme de croissance *Blossom* est le suivant :

```
function TN = createlist(nodesize,nodenum,dist)

% Ajouter le tenseur central
TN = -1*ones(1, nodesize) ;

NetSize = 1 ; %NetSize gives the number of tensor in the network

% Each iteration of the for loop adds a new layer of tensors with distance i
for i = 1 :dist
    % Each iteration of the two for loops scans the list for open indices and adds
        a new tensor at these indices
    for j = 1 :size(TN,1)
        for k = 1 :size(TN,2)
            if TN(j,k)==-1
                % Checknode evaluates if a tensor is ajdacent or redundant
                [num,prev] = checknode(TN,j,k) ;
                if num == nodenum % If the tensor is adjacent
                    % The tensor's row number is added to the list
                    TN(j,k)=NetSize+1 ;
                    TN(prev,end)=NetSize+1 ;
                    % A new row is added to the list
                    NewNode = -1*ones(1,nodesize) ;
                    NewNode(1) = prev ; % Prev is the parent tensor
                    NewNode(end) = j ; % j is the adjacent tensor
                    TN = [TN ;NewNode] ;
                    % We increase the number of tensors in the network
                    NetSize = NetSize + 1 ;
                elseif num == nodenum+1 % If the tensor is adjacent
                    TN(j,k) = prev ;
                    TN(prev,end) = j ;
                elseif num < nodenum
                    % The tensor's row number is added to the list
                    TN(j,k)=NetSize+1 ;
                    % A new row is added to the list
                    NewNode = -1*ones(1,nodesize) ;
                    NewNode(1) = j ; % j is the parent tensor
                    TN = [TN ;NewNode] ;
                    % We increase the number of tensors in the network
                    NetSize = NetSize + 1 ;
                end
                [rnum,rprev]=rchecknode(TN,j,k) ;
```

```
                if rnum == nodenum
                   TN(NetSize,end)=rprev ;
                   TN(rprev,2)=NetSize ;
                end
            end
        end
    end
end
```

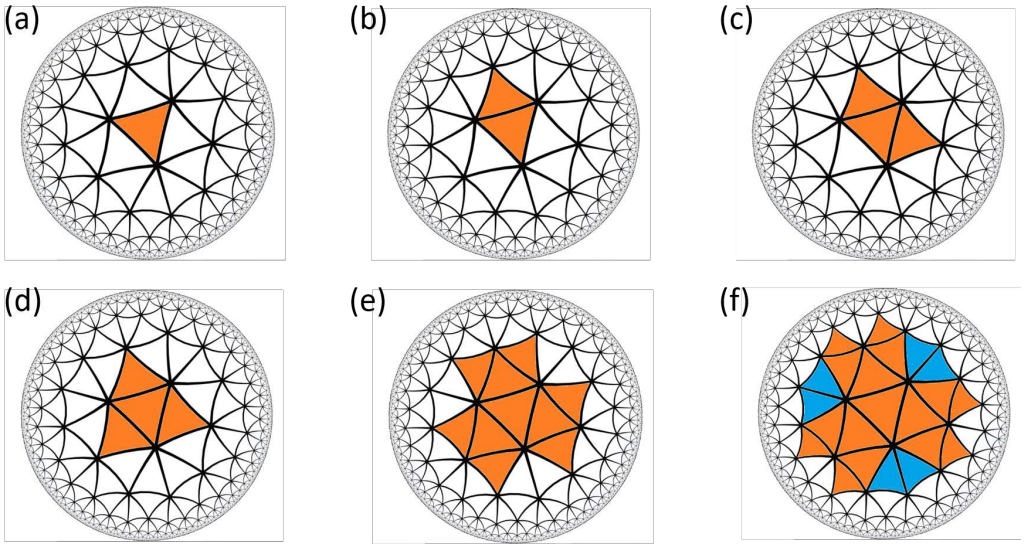


FIGURE 2.1 La construction d'un réseau utilisant l'algorithme Blossom, les tenseurs ajoutés étant oranges. Le réseau hyperbolique sert de guide. Dans l'étape (a), le tenseur central est placé. Les étapes (b)-(d) démontrent l'ajout d'un nouveau niveau dans l'algorithme. L'étape (e) montre la croissance d'un autre niveau. L'étape (f) montre un cas spécial qui se produit lors de l'ajout du prochain niveau. les tenseurs ajoutés (en bleu), sont adjacents l'un avec l'autre. C'est tenseurs sont identifiés avec le code *checknode*, qui est introduit plus loin.

La fonction *createlist* retourne la liste d'adjacence *TN* et prend en entrée les variables *nodesize*, *nodenum* et *dist*, qui sont définies comme suit :

- *nodesize* : Le nombre d'indices possédés par un seul tenseur. Dans le cas d'un dallage $\{n, m\}$, ce nombre est donné par n.

- *nodenum* : Le nombre de tenseurs autour d'un point dans le dallage. Il est représenté par m dans un dallage $\{n, m\}$.

- *dist* : La distance absolue entre un tenseur à la frontière du réseau et le tenseur central.

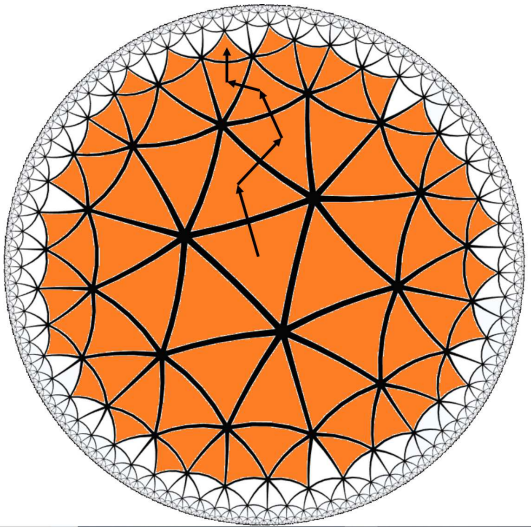
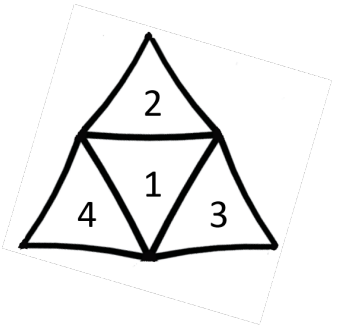


FIGURE 2.2 Le réseau, ici en orange, superposé sur l'espace hyperbolique complet. Les flèches montrent chaque niveau additionnel de tenseurs. Étant donné que l'on définit un code ne possédant que le tenseur central comme distance 0, les cinq niveaux supplémentaires nous donnent un code avec distance 5.

La première étape est donc de créer le tenseur central, et ainsi entamer la liste qui représente le réseau entier. Chaque itération de la boucle principale est définie comme étant la création de tous les tenseurs à une certain distance absolue du tenseur central. La troisième itération correspond donc à la création de tous les tenseurs avec distance 2 du tenseur central. Une matrice $N \times n$ est utilisée au lieu d'une matrice d'adjacence afin de conserver la mémoire, demandant $O(N)$ au lieu de $O(N^2)$. Ce résultat pourrait également être atteint avec une matrice éparse, mais un autre avantage, qui sera discuté plus en détail dans les prochaines sections, est qu'une telle liste permet d'accorder une orientation à chaque tenseur. Chaque tenseur possède un numéro qui équivaut à sa rangée dans la matrice. Pour une rangée donnée, chaque colonne correspond à un indice du tenseur. Toute la même colonne représente toujours le même indice pour tous les tenseurs, et la valeur dans la colonne est le numéro du tenseur attaché à cet indice. Comme convention, nous prenons que le tenseur en première position est le tenseur créé au cours du pas précédent, dorénavant appelé tenseur d'origine. La deuxième colonne est occupée par le tenseur qui est adjacent au premier en suivant le sens horaire. La figure 2.3 montre explicitement comment les tenseurs sont numérotés. Quand un tenseur possède un indice ouvert (qui n'est pas connecté à un autre tenseur), la colonne correspondante contient la valeur -1.

Une fois le tenseur central créé, et pour chaque itération suivante, l'algorithme parcourt la liste, créant un nouveau tenseur quand -1 est rencontré dans une colonne. Quand un nouveau tenseur est ajouté, le -1 dans la rangée du tenseur d'origine est remplacé par le



1		2	3	4
2		1	-1	-1
3		1	-1	-1
4		1	-1	-1

FIGURE 2.3 Une visualisation de l'état de la matrice créée par l'algorithme Blossom après la première croissance suivant l'ajout du tenseur central. Le numéro de la rangée dans le tableau correspond au numéro du tenseur à gauche, avec la colonne dénotant le numéro de l'indice utilisé.

numéro du nouveau tenseur. Un nouveau tenseur appartient à une de trois catégories, illustrées dans la figure 2.4. Prenant pour exemple un tenseur avec trois indices, ce dernier peut être :

- Solitaire : Tous les indices du nouveau tenseur, mis à part le premier, sont libres. Le premier indice est dénoté par le numéro du tenseur du pas précédent. Donc, si le nouveau tenseur n émerge du tenseur 3, la rangée n se lit : $[3, -1, -1]$.

- Adjacent : Le nouveau tenseur est en contact avec son tenseur d'origine et un autre tenseur préexistant. Étant donné que le réseau entier est bâti en sens horaire, ce deuxième tenseur rentre dans la deuxième ou dernière colonne, selon s'il se situe en position horaire ou anti-horaire comparativement au tenseur d'origine. Un tenseur n avec un autre tenseur adjacent $n-1$ en position horaire par rapport au tenseur d'origine 3 aurait la rangée : $[3, n-1, -1]$.

- Superposé : Le nouveau tenseur n a déjà été créé. Dans un tel cas, une nouvelle rangée n'est pas ajoutée à la liste, mais le numéro du tenseur déjà existant remplace le -1 dans la rangée du tenseur d'origine. Le -1 correspondant dans la rangée n est également remplacé par le numéro du tenseur d'origine.

C'est afin de distinguer entre ces trois cas que la fonction *checknode* fût écrite. Son fonctionnement est simple : *checknode* compte le nombre de tenseurs autour d'un sommet (ici démarqué par le point rouge dans la figure 2.5). C'est en comparant ce nombre au nombre de tenseurs autour de tout sommet (*nodenum*) que nous pouvons déterminer la catégorie. Rappelons que ce nombre est une propriété intrinsèque d'un dallage. Si le nombre est inférieur à *nodenum*-1, le nouveau tenseur est solitaire. Si le nombre est égal à *nodenum*-1, le nouveau tenseur complète le nombre autour du sommet et doit donc être adjacent, et si le

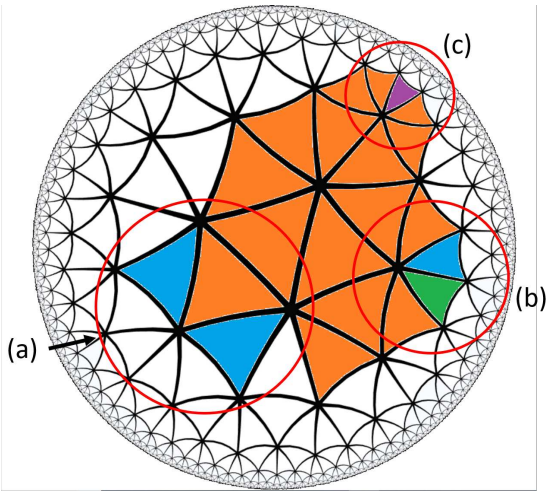


FIGURE 2.4 Un exemple des trois types de tenseurs. Les tenseurs colorés en orange sont des tenseurs déjà existants lors de l'ajout des nouveaux tenseurs. Dans le cas (a), les deux tenseurs bleus sont des tenseurs solitaires. Le cas (b) illustre un tenseur adjacent en vert, qui est ajouté après le tenseur solitaire en bleu. Le tenseur bleu est solitaire car, au moment de l'ajout, le tenseur adjacent n'existait pas. Le cas (c) dénote un tenseur superposé en mauve. Initialement, le tenseur mauve a émergé du tenseur orange à sa gauche en tant que tenseur adjacent. La superposition émerge quand le tenseur orange à droite essaie d'ajouter un tenseur à cette position également, créant la superposition.

nombre est égal à *nodenum*, le sommet est déjà saturé et le tenseur existe déjà. Le code de *checknode* est donné ici :

```
function [num,prev] = checknode(CN,j,k)
% Get number of tensors around a vertex
L = size(CN,2) ;
% Check if tensor is redundant, adjacent or solitary
num = 1 ; % num counts the number of iterations for the while loop
tk = k ; % tk keeps track of the column
tj = j ; % tj keeps track of the row
out = 0 ;
% The algorithm goes clockwise around a node until it finds the next open index.
while out ~= -1
    tk = mod(tk-1,L) ; % Wrap around to the next tensor around the vertex
    if tk == 0 % Fix the modulo problem
        tk = L ;
    end
    prev = tj ; % prev keeps the row in memory while we check for an open index
```

```
    out = CN(tj,tk) ;
    if out ~= -1
        tj = out ; % Go to new row
        tk = find(CN(tj, :)==prev) ; % Find appropriate column on new row
    end
    num = num+1 ; % Increase the number of tensors around the node.
end
```

Afin de compter le nombre de tenseurs autour d'un sommet, *checknode* utilise la liste de voisins comme une carte. Quant au sens de rotation, il est possible de constater que, le réseau étant bâti en sens horaire, un tenseur préexistant ou adjacent au nouveau tenseur se situe dans une position anti-horaire à ce dernier. Ainsi, faire un tour du sommet en sens horaire devrait faire le chemin complet autour du sommet et donc permettre de compter le nombre de tenseurs présents. La figure 2.5 représente cette constatation. Étant donné qu'un tenseur est ajouté après la vérification du nombre total de tenseurs autour d'un point, le décompte est fait à partir du tenseur d'origine. Nous pouvons suivre l'exemple donné dans la figure 2.5.

(1) Nous trouvons le numéro du prochain tenseur autour du sommet en regardant la colonne à gauche de celle où se trouve le nouveau tenseur. En regardant à gauche, nous effectuons une rotation en sens anti-horaire. Étant donné que le code est programmé dans MATLAB, nous devons ajouter une ligne pour nous assurer que si nous regardons à gauche de la première colonne, nous arrivons à la dernière colonne.

(2) Une fois le prochain tenseur trouvé, nous gardons en mémoire le numéro du tenseur actuel, puis nous allons à la rangée du prochain tenseur. Là, nous retrouvons la colonne contenant le tenseur que nous venons de quitter. Nous augmentons également le compte du nombre de tenseurs de 1.

(3) Nous répétons l'étape 1 et 2 jusqu'à temps que l'étape 1 retourne -1. Cela veut dire que nous sommes arrivés au dernier tenseur, et nous avons le nombre final de tenseurs autour du sommet.

Retournant au code de croissance *Blossom*, nous voyons que chaque résultat retourne le type de tenseur correspondant, traitant dans l'ordre suivant les tenseurs adjacents, superposés et solitaires. Finalement, le code contient une dernière fonction, soit *rchecknode*. Cette fonction est la même que *checknode*, mais vérifie les tenseurs en sens horaire. Cette fonction existe pour assurer que le premier tenseur ajouté dans un pas et le dernier tenseur ne sont pas le même ou adjacents, car lors de la création du premier tenseur le dernier n'existait pas

pour effectuer la vérification. Ces fonctions sont présentes universellement afin de prendre en compte des situations imprévues par l'auteur.

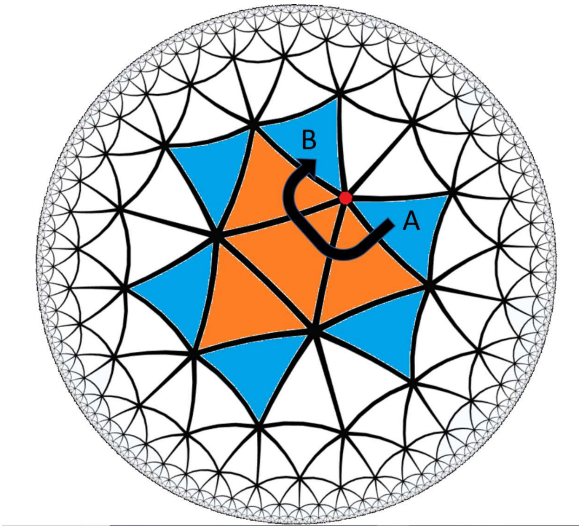


FIGURE 2.5 Le processus pour vérifier le nombre de tenseurs autour d'un sommet (ici dénoté en rouge). L'algorithme part du tenseur A et compte le nombre de tenseurs en sens horaire autour du sommet, jusqu'à temps qu'il arrive au dernier tenseur, dénoté B. Dans cette instance, le nombre de tenseurs est de 5, ce qui veut dire que le nouveau tenseur (A) est solitaire. Un résultat de 7 aurait déterminé que A est un tenseur adjacent, et 8 retournait un tenseur superposé.

2.1.2 Algorithme Concentrique

La deuxième méthode de création du réseau est celle dénommée croissance concentrique. Le nom provient du fait que chaque itération crée une couche complète de tenseurs autour de la couche précédente, comme l'illustre la figure 2.6. Les tenseurs dans la même couche peuvent avoir des distances absolues différentes les unes des autres, mais chaque pas crée un anneau complet.

L'algorithme de croissance concentrique est le suivant :

```
function TN = createcirclist(nodesize,nodenum,dist)

% Add central tensor
TN = -1*ones(1, nodesize) ;
NetSize = 1 ; % NetSize gives the number of tensor in the network
```

```
% Each iteration adds a new layer of tensors at distance i
for i = 1 :dist
    % Find out how many new tensors will be created
    NodeList = find(TN==-1) ;
    L = max(size(NodeList)) ;
    PrevNetSize = NetSize ;
    % Create an empty list for the new layer
    TNadd = -1*ones(L*(nodenum-2),nodesize) ;
    % Each iteration saturates an open vertex
    for j = 1 :L
        % Find the parent node for the fist new tensor around the vertex
        sourcenode = mod(NodeList(j),PrevNetSize) ;
        if sourcenode == 0
            sourcenode = PrevNetSize ;
        end
        NetSize = NetSize + 1 ;
        pos = NetSize - PrevNetSize ;
        % Adds the first new tensor around the vertex
        if j == 1
            TNadd(pos,1) = sourcenode ;
            TNadd(pos,2) = PrevNetSize + L*(nodenum-2) ; % If the tensor is the
                first of the new layer, adds the number of the last tensor of the
                new layer in the appropriate position
            TNadd(pos,end) = NetSize + 1 ;
            TN(NodeList(j)) = NetSize ;
        else
            TNadd(pos,end) = sourcenode ;
            TNadd(pos,1) = NetSize - 1 ;
            TNadd(pos,end-1) = NetSize + 1 ;
            TN(NodeList(j)) = NetSize ;
        end
        % Saturates the vertex by adding the subsequent tensors in a clockwise
            order
        for k = 2 :nodenum-2
            NetSize = NetSize + 1 ;
            pos = NetSize - PrevNetSize ;
            TNadd(pos,1) = NetSize - 1 ;
            if j == L && k == nodenum - 2
                TNadd(pos,end) = PrevNetSize + 1 ;
            else
                TNadd(pos,end) = NetSize + 1 ;
            end
```

```
        end
    end
    TN = [TN ;TNadd] ;
end
```

Cet algorithme possède plusieurs éléments en commun avec l'algorithme *Blossom*. Comme ce dernier, le réseau est représenté sous forme de liste, et l'orientation de chaque tenseur demeure la même avec les indices ouverts dénotés par -1. Le tenseur initial est également le point de départ, et les tenseurs sont ajoutés en sens horaire. Les variables prises en entrées sont définies de la même manière, avec l'exception de *dist*. Ici, la distance sera définie comme le numéro de la couche sur laquelle un tenseur est situé, commençant avec le tenseur central sur la couche 1.

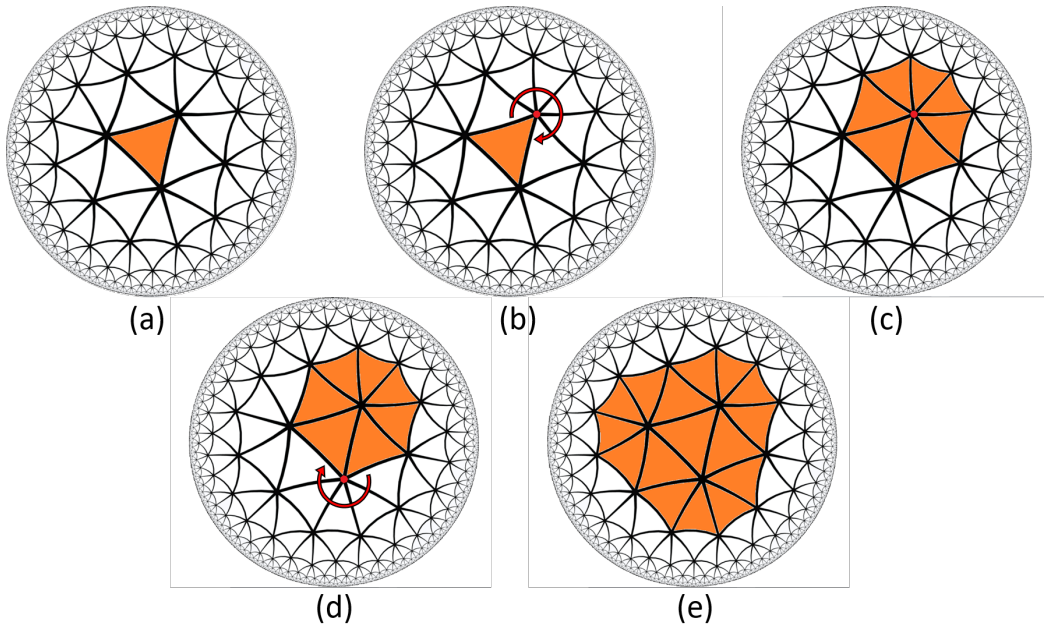


FIGURE 2.6 Les différentes étapes de l'algorithme de croissance concentrique. L'étape (a) est l'ajout du tenseur central. L'étape (b) détermine combien de tenseurs doivent être ajoutés autours du sommet en rouge. L'étape (c) illustre l'ajout de ces tenseurs, suivit de l'étape (d) qui est une répétition de l'étape (b) sur le sommet suivant. L'étape (e) montre la complétion du résultat, avec la complétion de la couche.

Afin d'expliquer la logique derrière ce code, regardons plus en détails ce qui se passe dans la figure 2.6. Comme nous l'avons expliqué précédemment, chaque sommet dans le réseau est entouré d'un certain nombre de tenseurs. L'algorithme de croissance concentrique sature chaque sommet appartenant à la couche $n-1$ avec les tenseurs appartenant à la couche n.

L'algorithme trouve d'abord les indices ouverts du réseau avec la fonction *find*(). Il est possible de voir que le nombre d'indices ouverts correspond graphiquement au nombre de côtés ouverts sur le réseau, et donc au nombre de sommets non saturés. Étant donné que chaque côté possède deux sommets, nous choisissons d'identifier un côté avec le sommet situé dans la direction de construction du réseau.

Une fois que le nombre de pas à faire pour construire une couche est déterminé, nous commençons immédiatement la construction de la nouvelle couche. Pour ce faire, nous créons une matrice *TNadd* dans laquelle nous allons bâtir cette nouvelle couche. La création de la liste d'adjacence de la couche sera sous-divisée encore plus en ajoutant les tenseurs autour de chaque sommet. On détermine ensuite le tenseur de la couche précédente adjacent à ce sommet, qu'on dénote *sourcenode*. On ajoute d'abord le premier tenseur de la nouvelle couche, puis nous ajoutons ensuite tous les autres tenseurs autour du sommet en sens horaire, à l'exception du dernier tenseur, qui est connecté également à la couche précédente. Ce tenseur sera rajouté lors du prochain pas, quand nous rajouterons les tenseurs autour du sommet suivant.

Quand le premier tenseur autour du sommet est ajouté, la première colonne dans son entrée contient le numéro du tenseur de la couche précédente auquel il est adjacent. La deuxième colonne contient le nombre du tenseur dans le sens anti-horaire (le dernier tenseur ajouté au sommet précédent), et la dernière colonne le nombre du tenseur dans le sens horaire (le tenseur suivant). Quand un tenseur qui n'est pas adjacent à un tenseur de la couche précédente est ajouté dans la liste d'adjacence, le nombre du tenseur précédent est ajouté dans la première colonne et le nombre du tenseur suivant est ajouté dans la dernière colonne.

Quand nous ajoutons le dernier tenseur dans la couche, la dernière colonne de son entrée dans la liste d'adjacence est le numéro du premier tenseur dans la couche. Le numéro du dernier tenseur doit également être ajouté dans la deuxième colonne du premier tenseur de la couche, concluant l'ajout d'une nouvelle couche. Cette boucle est répétée jusqu'à temps que le nombre de couches désirées soit obtenu. Comme dans l'algorithme précédent, les fonctions sont gardées aussi générales que possible afin d'accommoder des situations et des dallages génériques.

2.1.3 Algorithme Géodésique

La troisième façon de bâtir le code est l'utilisation de la distance géodésique. Cette distance est calculée utilisant la métrique imposée par le plongement du dallage dans le

disque de Poincaré,

$$ds^2 = 4\frac{dr^2 + rd\phi^2}{(1-r^2)^2}, \tag{2.1}$$

avec les coordonnées polaires (r, ϕ), où $0 \leq r < 1$ et $0 \leq \phi < 2\pi$. Afin d'avoir un réseau fini, et puisque le disque de Poincaré est infini, il faut tronquer r à une distance r_c. Chaque tenseur doit être placé en calculant sa distance et sa position relativement au tenseur précédent, et l'algorithme arrête d'ajouter des tenseurs quand aucune position disponible ne répond à $r < r_c$. La vérification pour un tenseur adjacent ou superposé demeure similaire.

2.1.4 Comparaison qualitative des méthodes

Ces trois méthodes résultent en des réseaux distincts, chacun ayant son utilité. Commençons par regarder la dernière méthode, la croissance *géodésique*. Cette méthode a l'avantage de baser la distance entre deux points dans la surface sur une mesure intrinsèque à l'espace. Elle devrait être utilisée pour la construction de réseaux sur lesquels des calculs de correspondance AdS/CFT sont effectués, ainsi que pour rechercher en profondeur toute corrélation entre la géométrie de l'espace hyperbolique et le dallage. De plus, cette méthode permet la création rapide de graphiques plongés dans le disque de Poincaré. Cependant, cette méthode est de loin la plus difficile à programmer, et demande une connaissance plus poussée en relativité générale pour être correctement implémentée. Trouver des isométries similaires à celles présentes dans le réseau hyperinvariant d'Evenbly est également difficile, car l'inhomogénéité de la frontière fait en sorte qu'une isométrie trouvée à une certaine distance ne sera pas nécessairement présente dans une autre.

La deuxième méthode, soit la croissance *concentrique*, a l'avantage de posséder une structure similaire au MERA, qui permet de comparer les résultats obtenus avec ce type de réseau, qui possède des propriétés déjà bien connues. Cette structure rend la construction idéale pour trouver des isométries dans le réseau, et se prête très bien à la correction d'erreur. Finalement, cette méthode de construction mène à un ordre de contraction plus facile à déterminer, ce qui sera discuté dans la prochaine section. La faille de ce réseau est que les tenseurs sur son périmètre ne sont pas tous à la même distance géodésique du centre. Cette distance géodésique correspond à l'échelle d'énergie de la CFT qui apparaît lors de la troncature du réseau infini. Deux tenseurs sur le périmètre ayant deux distances géodésiques différentes indique que deux niveaux d'énergies de la CFT coexistent sur le périmètre du réseau. Ceci change le comportement des corrélateurs calculés à partir de la matrice de covariance.

En dernier, la croissance *Blossom*, qui inclut à la fois des propriétés de la croissance géodésique et de la croissance concentrique. De la croissance géodésique elle hérite d'une meilleure définition de la distance entre deux tenseurs, tandis que de la croissance concentrique elle obtient un sens plus intuitif d'un point de coupure ainsi que d'une plus grande facilité de programmation. Cependant, bien que cette approche soit la plus facile à comprendre, elle a le défaut de posséder les inconvénients des deux approches précédentes sans leurs avantages. Le réseau basé sur cet algorithme n'est pas idéal pour la construction d'isométries, comme pour la méthode géodésique, car différents agencements de tenseurs peuvent également être retrouvés à différentes échelles. Étant donné que la distance n'est pas une distance géodésique, certains tenseurs sur le même périmètre se retrouvent dans différentes échelles d'énergie, modifiant également le comportement des corrélateurs.

La méthode utilisée sera donc l'algorithme de croissance concentrique, car le but ultime du code est de créer un réseau sur lequel des isométries peuvent être trouvées aisément. Quant aux corrélateurs, bien qu'on doive s'attendre à certaines divergences du résultat obtenu par *Jahn et al.* [30], le comportement général devrait être le même.

2.2 Contraction du réseau

L'étape suivante de notre algorithme est la contraction du réseau basé sur la liste créée dans la dernière section. Ici nous mettons en lumière la deuxième phase de création d'une matrice de covariance, lors de laquelle le réseau est finalisé et calculé afin d'obtenir la matrice génératrice finale. Cette section est divisée en trois sous-sections. La première sert d'introduction au code utilisé pour accomplir les opérations de base sur les portes de parité, soit la contraction, l'auto-contraction et la permutation cyclique. La deuxième sous-section parle de l'orientation initiale donnée aux tenseurs, et répond à certaines des questions laissées sans réponse durant la création du réseau. La sous-section finale élabore sur la construction elle-même du réseau.

2.2.1 Opérations fondamentales

Trois opérations sont nécessaires pour construire et manipuler un réseau de tenseurs de type porte de parité, soit la contraction, l'auto-contraction et la permutation cyclique. L'ensemble de ces trois opérations permet d'effectuer une contraction arbitraire du réseau, soit d'un sous-ensemble donné ou du réseau entier. Ici, nous allons ajouter une contrainte supplémentaire dans la convention de signe à travers nos opérations, définie ici :

Définition 2.2.1 (Contrainte d'uniformité)**.** Les opérations fondamentales sont définies pour qu'une matrice A antisymétrique ait un élément $A_{i,j}$ positif si $i < j$ et négatif si $i > j$.

Ceci a été établi majoritairement dans le but de produire des résultats facilement vérifiables, particulièrement dans les prochaines sections, et est une contrainte qui est également respectée dans les résultats de l'article de référence [30].

Contraction

La contraction est une opération essentielle pour le calcul de n'importe quel réseau. Dans le cas de portes de parité, cette opération demeure très simple, car elle revient à deux produits matriciels. Fondamentalement, deux portes de parité de rang n avec matrices génératrices A et B respectivement se contractent pour former une porte de parité de rang $2n-2$ avec une matrice génératrice C de la manière suivante :

$$C = \begin{bmatrix} \begin{pmatrix} A_{1,1} & \cdots & A_{1,n-1} \\ \vdots & \ddots & \vdots \\ A_{n-1,1} & \cdots & A_{n-1,n-1} \end{pmatrix} & \begin{pmatrix} A_{1,n} \\ \vdots \\ A_{n-1,n} \end{pmatrix} \begin{pmatrix} B_{1,2} & \cdots & B_{1,n} \end{pmatrix} \\ -\begin{pmatrix} B_{1,2} \\ \vdots \\ B_{1,n} \end{pmatrix} \begin{pmatrix} A_{1,n} & \cdots & A_{n-1,n} \end{pmatrix} & \begin{pmatrix} B_{2,2} & \cdots & B_{2,n} \\ \vdots & \ddots & \vdots \\ B_{n,2} & \cdots & B_{n,n} \end{pmatrix} \end{bmatrix}$$

Dans ce cas-ci, nous contractons le dernier indice de la matrice A avec le premier indice de la matrice B. La preuve pour cette équation est faite par Alexander Jahn et al [30], ainsi que par Bravyi [32]. Ce résultat est intuitif, car C combine les corrélateurs affectés par la contraction, comme illustré dans la figure 2.7. Les corrélateurs qui ne sont pas impliqués dans la contraction demeurent inchangés, car leurs variables de Grassmann correspondantes restent complètement dans l'ensemble initial Φ_A ou dans Φ_B. Les corrélateurs dont les variables de Grassmann sont affectées deviennent des nouveaux corrélateurs qui sont simplement le produit des anciens. Par exemple, $C_{i,j}$, le corrélateur entre ϕ_i dans Φ_A et ϕ_j dans Φ_B sera égal à $A_{i,n}B_{1,j}$.

Le test final est de confirmer la condition 2.2.1, soit que $C_{i,j} < 0$ pour $i > j$ (par la nature des matrices antisymétriques, la satisfaction de cette version de la contrainte remplie la

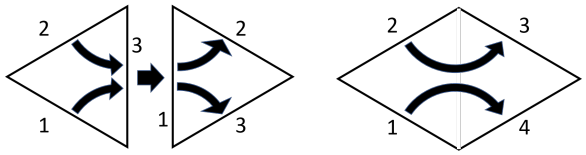


FIGURE 2.7 La combinaison des corrélateurs lors de la contraction. Chaque flèche représente un corrélateur, avec l'orientation indiquée et une valeur de 1.

contrainte complète). Ceci est fait aisément en regardant la multiplication de matrices qui composent C. Nous savons que la contrainte est respectée pour la partie supérieure gauche et la partie inférieure droite de C, car ces blocs sont les matrices partielles de A et de B, qui respectent déjà la contrainte 2.2.1. Pour la partie supérieure droite (et par asymétrie la partie inférieure gauche), la contrainte est remplie si $A_{i,n}B_{1,j} > 0$. Étant donné que $A_{i,n} > 0$ et que $B_{1,j} > 0$, le produit des deux est bien sûr positif, remplissant la contrainte. Le code pour la contraction est le suivant :

```
function C = tcontract(A,B)

% Contracts the last index of tensor A with the first index of tensor B to return
    tensor C
C = [A(1 :end-1,1 :end-1)
    A(1 :end-1,end)*B(1,2 :end) ;-1*B(1,2 :end).'*A(1 :end-1,end).'
    B(2 :end,2 :end)] ;
```

L'entrée du code est donc les deux tenseurs A et B, et l'algorithme retourne la matrice contractée C, présumant que le dernier indice de A est contracté avec le premier indice de B. Bien que ce code perd un peu de généralité par lui-même, il est toujours possible d'effectuer une permutation des indices afin d'orienter les deux indices contractés dans ces positions.

Auto-contraction

L'auto-contraction est une autre opération de base, et demeure presque aussi essentielle que la contraction. Dans le contexte de l'algorithme de contraction du réseau, qui sera introduit dans la section 2.2.3, les auto-contractions sont nécessaires dans deux cas principaux : la contraction d'un tenseur qui est connecté au réseau par deux indices ou plus, ou toute contraction de dimension $\chi > 2$. Ces deux situations reviennent au même quand on considère que le deuxième cas est similaire à la contraction de deux tenseurs superposés. Bien que l'équation d'auto-contraction est démontrée par Alexander Jahn et al. dans une

publication antérieure [30], nous allons refaire la preuve ici pour spécifier une remarque importante manquante dans leur preuve. Pour l'algorithme d'auto-contraction du tenseur avec matrice génératrice A, nous supposons que les deux premiers indices sont contractés ensemble. Encore une fois, cela peut être généralisé à la contraction de n'importe quels deux indice adjacents du tenseur par permutation cyclique. Quant à l'auto-contraction de deux indice non-adjacents, ce cas n'est pas pertinent dans ce contexte et donc n'est pas traité dans la preuve ou dans le code.

Commençons avec la preuve pour la formule finale. Le tenseur initial est un tenseur $T(\Phi)$ de dimension n donné par

$$T(\Phi) = c \exp\left\{\frac{1}{2}\Phi^T A \Phi\right\}. \tag{2.2}$$

À partir d'ici, nous allons ignorer le facteur c, qui sera redéfini à la fin de cette preuve, et seulement traiter l'exponentielle. Exprimant l'exponentielle par $\exp\left\{\sum_{i=1}^{d-1}\sum_{j=i+1}^{d} A_{i,j}\phi_i\phi_j\right\}$, nous contractons les deux premiers indices par l'entremise d'une intégration de Grassmann ($T(\Phi)_c$ dénote ici le tenseur contracté) :

$$\begin{aligned} T(\Phi)_c &= c \int \mathrm{d}\phi_1 \int \mathrm{d}\phi_2 \exp\left\{A_{1,2}\phi_1\phi_2 + \sum_{i=1}^{d-1}\sum_{j=i+1}^{d} A_{i,j}\phi_i\phi_j\right\} \\ &= c \exp\left\{\sum_{i=3}^{d-1}\sum_{j=i+1}^{d} A_{i,j}\phi_i\phi_j\right\} \int \mathrm{d}\phi_1 \int \mathrm{d}\phi_2 \exp\left\{(1+A_{1,2})\phi_1\phi_2 + \sum_{j=3}^{d}(A_{1,j}\phi_1\phi_j + A_{2,j}\phi_2\phi_j)\right\}. \end{aligned} \tag{2.3}$$

Développant l'exponentielle dans l'équation 2.3, on remarque qu'elle peut être simplifiée à

$$\begin{aligned} \exp\left\{(1+A_{1,2})\phi_1\phi_2 + \sum_{j=3}^{d}(A_{1,j}\phi_1\phi_j + A_{2,j}\phi_2\phi_j)\right\} &= 1 + (1+A_{1,2})\phi_1\phi_2 \\ &\quad + \sum_{j=3}^{d}(A_{1,j}\phi_1\phi_j + A_{2,j}\phi_2\phi_j) \\ &\quad + \left(\sum_{i=3}^{d-1}\sum_{j=i+1}^{d}(A_{i,1}A_{2,j} - A_{i,2}A_{1,j})\phi_1\phi_2\phi_i\phi_j\right), \end{aligned}$$

grâce à l'équation 1.12. Le deuxième terme est non nul une fois intégré, ainsi que le quatrième terme. Il est donc possible d'exprimer l'équation :

$$\begin{aligned} T(\Phi)_c &= c\exp\left\{\sum_{i=3}^{d-1}\sum_{j=i+1}^{d} A_{i,j}\phi_i\phi_j\right\}\left(1 + A_{1,2} + \sum_{i=3}^{d-1}\sum_{j=i+1}^{d}(A_{i,1}A_{2,j} - A_{i,2}A_{1,j})\phi_i\phi_j\right) \\ &= c(1+A_{1,2})\exp\left\{\sum_{i=3}^{d-1}\sum_{j=i+1}^{d} A_{i,j}\phi_i\phi_j\right\}\left(\sum_{i=3}^{d-1}\sum_{j=i+1}^{d}\frac{A_{i,1}A_{2,j} - A_{i,2}A_{1,j}}{1+A_{1,2}}\phi_i\phi_j\right). \end{aligned} \tag{2.4}$$

Nous devons être prudents ici. Bien que $A_{i,j} > 0$ si $i < j$, et donc $1 + A_{1,2}$ ne devrait jamais atteindre 0, une autre définition peut causer des erreurs quand $A_{1,2} \to -1$. Ceci est dû au fait que le code ne prend pas en compte le facteur c devant l'exponentielle, mais traite celui-ci séparément du reste de l'équation. Le terme en $(1 + A_{1,2})^{-1}$ va donc diverger et retourne un résultat invalide. Lors des premiers tests avec le code, quand une autre contrainte sur A était utilisée, cette erreur n'était pas initialement apparente. Continuant la preuve, on obtient

$$\begin{aligned} T(\Phi)_c &= c(1+A_{1,2})\exp\left\{\sum_{i=3}^{d-1}\sum_{j=i+1}^{d} A_{i,j}\phi_i\phi_j\right\}\exp\left\{\sum_{i=3}^{d-1}\sum_{j=i+1}^{d}\frac{A_{i,1}A_{2,j} - A_{i,2}A_{1,j}}{1+A_{1,2}}\phi_i\phi_j\right\} \\ &= c(1+A_{1,2})\exp\left\{\sum_{i=3}^{d-1}\sum_{j=i+1}^{d}\left(A_{i,j} + \frac{A_{i,1}A_{2,j} - A_{i,2}A_{1,j}}{1+A_{1,2}}\right)\phi_i\phi_j\right\}. \end{aligned} \tag{2.5}$$

La première ligne de 2.5 peut être expliquée par le fait que

$$\begin{aligned} \left(\sum_{i=3}^{d-1}\sum_{j=i+1}^{d}(A_{i,1}A_{2,j} - A_{i,2}A_{1,j})\phi_i\phi_j\right)^2 &= \left(\sum_{i=3}^{d}\sum_{j=3}^{d} A_{i,1}A_{2,j}\phi_i\phi_j\right)^2 \\ &= \left(\sum_{i=3}^{d} A_{i,1}\phi_i\right)^2\left(\sum_{i=3}^{d} A_{2,i}\phi_i\right)^2 = 0, \end{aligned} \tag{2.6}$$

dû à la propriété fondamentale des variables de Grassmann, qui stipule que $\phi_i^2 = 0$. Cette preuve se généralise aux puissances supérieures à 2, justifiant la première ligne de l'équation 2.5. La dernière ligne de 2.5 donne une équation très similaire à celle du tenseur initial $T(\Phi)$, et nous exprimons $T(\Phi)_c$ dans la même forme : $T(\Phi)_c = c_c \exp\{(\frac{1}{2}\Phi_c^T A_c \Phi_c)\}$. Ici, nous suivons les définitions

$$\phi_{c,i} = \phi_{i+2}, \tag{2.7}$$

$$A_{c,i,j} = A_{i+2,j+2} + \frac{A_{i+2,1}A_{2,j+2} - A_{i+2,2}A_{1,j+2}}{1+A_{1,2}}, \tag{2.8}$$

$$c_c = c(1+A_{1,2}). \tag{2.9}$$

Ce résultat est intuitif quand nous observons graphiquement l'auto-contraction. Regardant l'exemple illustré dans la figure 2.8, nous voyons que les nouveaux corrélateurs sont une combinaison linéaire du produit des corrélateurs originaux. Ces corrélations seront qualifiées de secondaires, car elle sont ajoutées au corrélateur initial. Le poids associé à cet ajout est donc $1 + A_{1,2}$, ce qui est logique.

Le code traitant cette auto-contraction est le suivant :

```
function C = selfcontract(C)

%Contract first two indices
NewC = zeros(size(C)-2) ;
for k = 3 :length(C)
    for p = 3 :length(C)
        NewC(k-2,p-2) = C(k,p) + (C(k,1)*C(2,p)-C(k,2)*C(1,p))/(1+C(1,2)) ;
    end
end

C = NewC ;
```

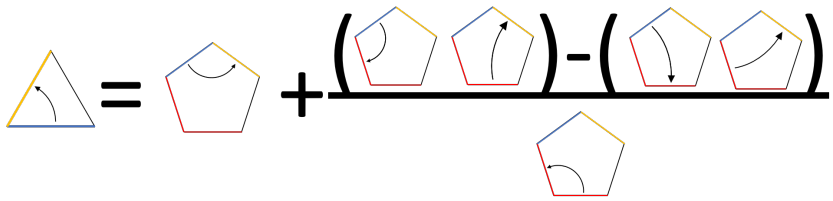


FIGURE 2.8 Une représentation visuelle de l'élément $(1,2)$ de la matrice après la contraction des deux premiers indices. Les éléments impliqués sont représentés par une flèche. Les deux arêtes en rouge correspondent aux indices contractés, et les arêtes en bleu et en jaune sont les indices 3 et 4 respectivement, qui deviendront les indices 1 et 2 après l'auto-contraction.

Comme nous pouvons le voir, ce code est très bref, prenant en entrée la matrice C de dimension $n \times n$ et retournant une matrice C de dimension $(n-2) \times (n-2)$. Le code crée une nouvelle matrice à dimension réduite, puis calcule en boucle chaque élément dans la matrice selon la formule trouvée dans cette section.

Comme mentionné plus haut, les deux premiers indices sont contractés, mais n'importe quelle paire d'indices adjacents peut être utilisée par permutation cyclique. La généralisation

à deux indices non-adjacents est possible mais beaucoup plus compliquée et ne sera pas résolue ici.

Rotation

Cette opération est cruciale dans la généralisation des deux opérations précédentes, permettant de réorienter les tenseurs afin de remplir les contraintes de la contraction (dernier indice avec premier indice) et de l'auto-contraction (deux premiers indices). Cette permutation cyclique des sites fermioniques est effectivement un re-étiquetage des indices du tenseur, et en langage informatique l'opération effectuée pour faire cette permutation est dénommée *décalage circulaire*. Les deux termes seront interchangeables dans le contexte du texte suivant, un dénotant l'action physique et l'autre l'algorithme informatique correspondant. Cette opération est illustrée dans la figure 2.9. On note ici que le site 1 avant la rotation devient le site 2 après la rotation, voulant dire que l'arête correspondante reste à la même place mais l'étiquette change. Ceci n'est pas intuitif, car cela voudrait dire qu'un tenseur à priori invariant par rotation n'aurait pas la même matrice génératrice avant une rotation, qu'après. Regardons les effets d'une permutation cyclique plus concrètement. Dans l'équation suivante, on dénote la permutation de n sites sur le tenseur T par $\sigma_i(T(\Phi))$, qui décale chaque site par i éléments. Une valeur négative de i indique une permutation cyclique en sens anti-horaire. En prenant l'exemple d'un tenseur typique d'un dallage 3,7, on exprime $\sigma_i(\Phi)$ l'application de la rotation directement sur la matrice génératrice, donnant

$$\begin{aligned}\sigma_2(\phi_1\phi_2 + \phi_1\phi_3 + \phi_2\phi_3) &= \phi_2\phi_3 + \phi_2\phi_1 + \phi_3\phi_1 \\ &= -\phi_1\phi_2 - \phi_1\phi_3 + \phi_2\phi_3.\end{aligned}$$

En termes "gauche-droite", nous définissons une permutation en sens horaire (anti-horaire) comme un décalage vers la gauche (droite). L'exemple précédent serait donc un décalage de 2 vers la gauche. On peut voir grâce à cette équation que la permutation cyclique d'un tenseur gaussien donne

$$\sigma_i(T(\Phi)) = \sigma_i\Big(c\exp\Big\{\frac{1}{2}\Phi^T A\Phi\Big\}\Big) = c\exp\Big\{\frac{1}{2}\Phi^T\sigma_i(A)\Phi\Big\} \tag{2.10}$$

La nouvelle matrice génératrice possède les mêmes éléments que la matrice avant la permutation, mais l'élément (k,l) est remplacé par l'élément $(k+i, l+i)$, le tout modulo n, avec n la longueur du vecteur Φ. En reprenant l'exemple plus haut, une permutation cyclique dans le sens horaire de magnitude 3 change la matrice génératrice A de cette manière :

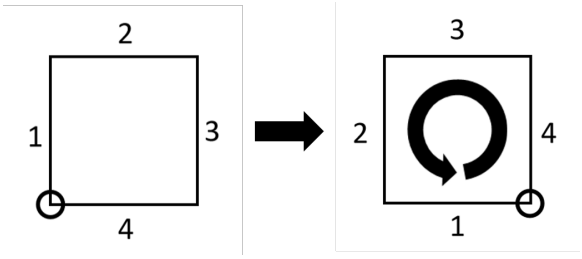


FIGURE 2.9 Une représentation d'une rotation anti-horaire des indices d'un tenseur de rang 4. Le cercle indique où nous commençons l'étiquetage des indices.

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 \\ -1 & -1 & -1 & 0 \end{bmatrix} \rightarrow \sigma_3(A) = \begin{bmatrix} 0 & -1 & -1 & -1 \\ 1 & 0 & 1 & 1 \\ 1 & -1 & 0 & 1 \\ 1 & -1 & -1 & 0 \end{bmatrix}.$$

C'est ici que nous pouvons voir le problème potentiel. Cette permutation ne respecte pas la contrainte 2.2.1. De plus, une rotation de ce type avant une auto-contraction pourrait causer l'erreur $A_{1,2} = -1$, qui comme l'a été expliqué auparavant fait diverger la valeur obtenue par l'équation 2.5 en annulant le dénominateur. Que faire dans ce cas ? Pour résoudre ceci, il faut se rappeler que le système simulé ici est un système de spins. Chaque permutation des degrés de liberté des spins affecte une transformation de Jordan-Wigner différente, modifiant le résultat précédent. Étant donné que les cas étudiés jusqu'à maintenant peuvent être représentés par des états dimères, nous allons effectuer une transformation dans la base des opérateurs de Majorana. Commençons par définir un opérateur de permutation dans la base des spins S_σ, avec $i \rightarrow \sigma(i) = \sigma_i$.

$$\begin{aligned} |\psi'\rangle = S_\sigma |\psi\rangle &= S_\sigma \sum_{j \in \{0,1\}^{\times N}} T_{j_1, j_2, \cdots, j_N} |j_1, j_2, ..., j_N\rangle \\ &= \sum_{j \in \{0,1\}^{\times N}} T_{j_1, j_2, \cdots, j_N} |\sigma(j_1), \sigma(j_2), ..., \sigma(j_N)\rangle \\ &= \sum_{j \in \{0,1\}^{\times N}} T'_{j_1, j_2, \cdots, j_N} |j_1, j_2, ..., j_N\rangle , \end{aligned} \tag{2.11}$$

où

$$T'_{j_1,j_2,\cdots,j_N} = T_{\sigma^{-1}(j_1),\sigma^{-1}(j_2),\cdots,\sigma^{-1}(j_N)}. \tag{2.12}$$

Les opérateurs de Majorana se transforment comme $\gamma_k \to S_\sigma \gamma_k S_\sigma^\dagger$. Posant $k = -1$, nous avons la transformation suivante :

$$\begin{aligned}
\gamma_1 &= X_1 \to X_2 = Z_1\gamma'_3 \\
\gamma_2 &= Y_1 \to Y_2 = Z_1\gamma'_4 \\
\gamma_3 &= Z_1X_2 \to Z_2X_3 = Z_1\gamma'_5 \\
\gamma_4 &= Z_1Y_2 \to Z_2Y_3 = Z_1\gamma'_6 \\
&\vdots \\
\gamma_{2N-1} &= Z_1...Z_{N-1}X_N \to X_1Z_2...Z_N = -Z_1\gamma'_1P_{tot} \\
\gamma_{2N} &= Z_1...Z_{N-1}Y_N \to Y_1Z_2...Z_N = -Z_1\gamma'_2P_{tot}
\end{aligned} \tag{2.13}$$

où $P_{tot} = Z_1...Z_N$ est l'opérateur de parité de l'état entier et a une valeur propre de 1 pour les états pairs. De ces équations, nous pouvons voir que ces opérateurs de Majorana ne sont pas les mêmes que ceux définis par une transformation de Jordan-Wigner. Les équations 2.14 montrent l'impact de cette transformation sur la parité des dimères.

$$\begin{aligned}
(\gamma_j + ip_{j,k}\gamma_k)\,|\psi\rangle &\to S_{+1}(\gamma_j + ip_{j,k}\gamma_k)S_{+1}^\dagger S_{+1}\,|\psi\rangle \\
&= (S_{+1}\gamma_j S_{+1}^\dagger + ip_{j,k}S_{+1}\gamma_k S_{+1}^\dagger)\,|\tilde{\psi}\rangle \\
&= \begin{cases}
Z_1(\tilde{\gamma}_{j+2} + ip_{j,k}\tilde{\gamma}_{k+2})\,|\tilde{\psi}\rangle & \text{si} \quad j,k < 2N-1 \\
Z_1(\tilde{\gamma}_{j+2} + ip_{j,k}\tilde{\gamma}_{k+2-2N}P_{tot})\,|\tilde{\psi}\rangle & \text{si} \quad j < 2N-1, k \geq 2N-1 \\
-Z_1(\tilde{\gamma}_1 + ip_{j,k}\tilde{\gamma}_2)P_{tot}\,|\tilde{\psi}\rangle & \text{si} \quad j = 2N-1, k = 2N
\end{cases}
\end{aligned} \tag{2.14}$$

Ceci fait en sorte que la matrice génératrice, avec une permutation effectuée dans la base des spins, demeure invariante par permutation cyclique si le tenseur est pair. Si le tenseur est impair, la symétrie de rotation n'est pas préservée, et une chaîne d'opérateurs Z est créée le long de la rotation. Cette réalité est mieux illustrée en regardant des diagrammes de dimères, tels ceux dans la figure 2.10.

Ici nous notons rapidement que deux chemins sont possibles, soit la rotation horaire et

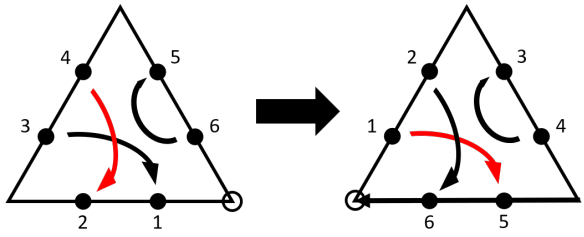


FIGURE 2.10 Une rotation horaire d'amplitude 1. On voit que cette rotation change la parité des dimères, avec $p_{2,4} = -1 \rightarrow p_{2,6} = 1$ et $p_{3,1} = 1 \rightarrow p_{1,5} = -1$

anti-horaire. Les deux états résultants sont équivalents avec un signe négatif global.

Choisissant une rotation dans l'espace des spins, une permutation cyclique ressemble maintenant à ceci :

$$A = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} \rightarrow \sigma_2(A) = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix},$$

ce qui mène au code utilisé.

```
function GamCorr = cycleperm(GamCorr,pnum)

% Converts clockwise permutation into its equivalent counterclockwise permutation.
if pnum<0
    L = size(GamCorr,1) ;
    pnum = L + pnum ;
end

% Permutes GamCorr in a counterclockwise direction pnum times
for k = 1 :pnum
    GamCorr =
        [-1*GamCorr(end,end),-1*GamCorr(end,1 :end-1) ;-1*GamCorr(1 :end-1,end)
        ,GamCorr(1 :end-1,1 :end-1)] ;
end
```

Nous prenons en entrée un tenseur *GamCorr* et la magnitude et direction de la per-

mutation *pnum*. Comme mentionné plus haut, une valeur positive (négative) correspond à une permutation en sens horaire (anti-horaire). Pour éviter des erreurs de signe, une permutation anti-horaire est transformée en permutation en sens horaire. La permutation est ensuite performée en répétant une permutation d'un site plusieurs fois.

2.2.2 Orientation initiale

L'orientation initiale des tenseurs fût une question persistante au cours de la création de l'algorithme. Cela est dû au fait que pour la majorité de la programmation, les permutations étaient effectuées avec la première méthode introduite dans la section précédente. Étant donné que ces permutations ne laissaient pas la matrice génératrice invariante, il était important d'orienter correctement les tenseurs avant de commencer la contraction. L'orientation de ces tenseurs doit être telle que toute rotation initiale est annulée par les permutations nécessaires pendant la contraction du réseau entier. Ceci est extrêmement difficile à visualiser, et est mieux décrit par la figure 2.11.

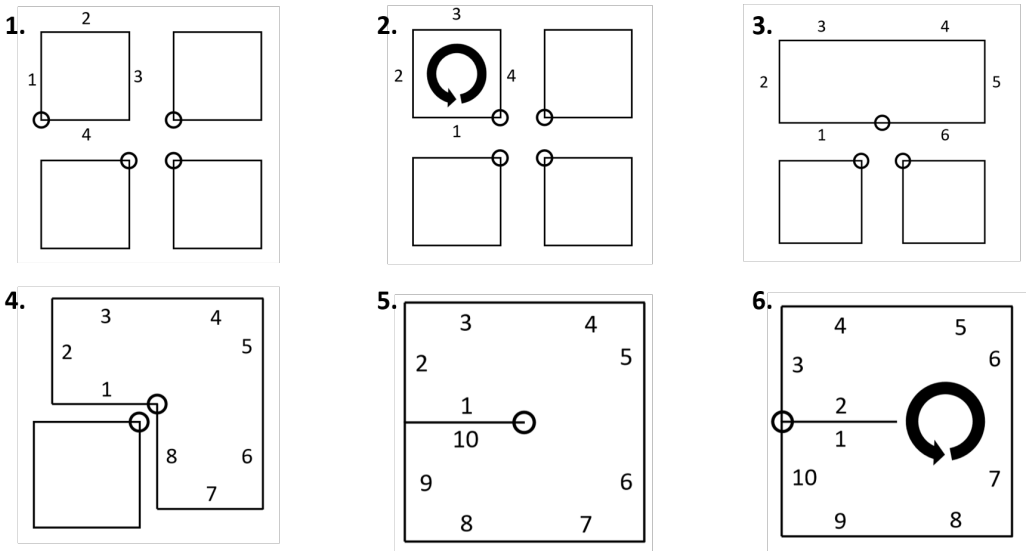


FIGURE 2.11 Une procédure d'orientation initiale afin de préserver la condition $A_{i,j} > 0 \forall i < j$. Les tenseurs de rang 4 sont contractés ensemble en sens horaire, avec les cercles spécifiant le début de l'étiquetage. La rotation faite lors de la deuxième étape existe pour compenser la rotation effectuée à la fin de la contraction totale. Afin d'avoir cette rotation dans l'étape 2, l'orientation du tenseur initial doit être choisie à ces fins lors de l'étape 1.

La liberté de permutation des tenseurs initiaux est fixée en imposant des contraintes sur la frontière du réseau. Dans ce mémoire, une condition d'anti-périodicité est imposée sur la frontière. C'est cette condition qui force la matrice génératrice A à avoir des éléments $A_{i,j}$

positifs pour $i < j$ et négatifs quand $i > j$. Afin de maintenir ces conditions sur la totalité du réseau, seul le tenseur central peut subir des permutations cycliques. Les tenseurs externes doivent donc déjà être dans la bonne position pour permettre la contraction, avec leur indice le plus bas aligné avec l'indice contracté en premier.

Bien que notre méthode de permutation cyclique garde les tenseurs externes invariants par permutations (s'ils sont pairs), nous avons quand même accommodé cette orientation initiale dans la création du réseau pour permettre l'utilisation du code avec des tenseurs sans spins.

2.2.3 Ordre de contraction

Grâce à l'algorithme de création du réseau de tenseurs, l'ordre de contraction du réseau est simple : il suit le même ordre que celui dans lequel les tenseurs ont été ajoutés à la liste du réseau. Cela est fait afin que chaque tenseur ajouté à la liste ait déjà le plus petit indice aligné avec le tenseur central, avec lequel il sera contracté. Cet algorithme, comme le reste du code, ne marche seulement qu'avec les tenseurs pairs, car les tenseurs impairs demandent un code qui peut traiter convenablement la chaîne d'opérateurs Z qui sont créés lors de la rotation du tenseur central. D'autres articles donnent de meilleurs aperçus de cette situation [44].

Le code utilisé pour la contraction du réseau de tenseurs complet est le suivant :

```
function [GMatrix,neigh] = Contractnet(TN,genmatrix,inirot)

% Build first tensor and neighbor list
GMatrix = cycleperm(genmatrix,inirot) ;
neigh = [TN(1,2 :end) TN(1,1)] ;

% Add tensors in the order they were built in createlist
L = size(TN,1) ;
for i = 2 :L
    pos = find(neigh == i) ;
    N = size(neigh,2) ;
    if size(pos,2)>1 %There will be a self-contraction
        %Check to ensure correct indice is contracted first
        if pos(end)~=N || pos(1)~=1
            cnum = N-pos(1) ;
```

```
            GMatrix = cycleperm(GMatrix,cnum) ;
            neigh = [neigh(pos(end) :end) neigh(1 :pos(1))] ;
        end
        % Contract in new tensor
        GMatrix = tcontract(GMatrix,genmatrix) ;
        neigh = [neigh(1 :end-1) TN(i,2 :end)] ;
        % Complete self-contraction (rotation to align + self-contract)
        GMatrix = cycleperm(GMatrix,1) ;
        GMatrix = selfcontract(GMatrix) ;
        neigh = neigh(2 :end-1) ;
    else
        cnum = N-pos ;
        GMatrix = cycleperm(GMatrix,cnum) ;
        neigh = [neigh(pos+1 :end) neigh(1 :pos)] ;
        % Contract in new tensor
        GMatrix = tcontract(GMatrix,genmatrix) ;
        neigh = [neigh(1 :end-1) TN(i,2 :end)] ;
    end
end
```

L'algorithme prend en entrée la matrice du réseau (la variable *TN*), créée lors de l'étape précédente, la matrice génératrice des tenseurs (la variable *genmatrix*), et la rotation initiale du tenseur central (la variable *inirot*). Le produit final est un réseau de tenseurs contracté (la variable *GMatrix*) et une liste de tenseurs adjacents au réseau (la viable *neigh*). Le fait qu'une seule matrice génératrice est prise en entrée est dû à la contrainte imposée par la correspondance AdS/CFT, qui exige que le réseau de tenseurs soit uniforme. Si l'utilisateur du code désire généraliser celui-ci, une solution suggérée serait d'ajouter une colonne à la liste du réseau. Cette colonne identifierait la matrice génératrice associée au tenseur, et l'entrée du code serait changé à un vecteur de matrices plutôt qu'une seule matrice génératrice, l'identifiant étant la position de la matrice génératrice voulue dans le vecteur. Finalement, la rotation du tenseur central n'est pas nécessaire pour le code, et originalement est un artefact des versions précédentes. Cette variable a été gardée dans le but d'éventuellement écrire un code plus général, incluant les tenseurs impairs. Cela dit, une autre fonction potentielle est l'agrandissement d'un réseau déjà contracté, dans quel cas cette rotation serait importante.

La première étape du code est la création du réseau initial, dénoté par la variable *GMatrix*, qui consiste en un tenseur central, ainsi qu'une liste de voisins *neigh* en ordre horaire. C'est de cette liste que nous allons déterminer le prochain tenseur à ajouter. Comme la liste du réseau est bâtie à partir de l'ensemble des voisins de chaque tenseur en sens horaire, la

construction de la liste des voisins du réseau est directe. Le premier voisin dans la liste du réseau devient le dernier dans la liste de voisins afin de maintenir la boucle dans le réseau.

La prochaine étape dans le code est la vérification de l'existence d'une auto-contraction. Ceci est accompli en regardant si le prochain tenseur à être contracté dans le tenseur central est connecté à celui-ci avec plus d'un indice. C'est fait en utilisant la fonction *find*() avec *neigh* en entrée et en évaluant la taille du résultat. Si elle est plus grande que 1, il existe une auto-contraction et une rotation est effectuée afin de contracter le bon indice en premier, qui demande, par la nature périodique de la liste des voisins, que le deuxième indice soit au début de la liste. Une fois que le premier indice est contracté, les contractions suivantes seront toutes des auto-contractions. Le deuxième indice étant déjà situé dans le premier indice, il ne suffit que de faire une permutation en sens anti-horaire pour amener les deux indice dans la bonne position. Cette suite de rotations suivie d'une auto-contraction est répétée jusqu'à temps que le tenseur soit complètement lié au réseau.

Si le tenseur n'est connecté qu'une fois au réseau central, le réseau effectue une permutation cyclique afin que l'indice correspondant soit dans le dernier indice quand nécessaire. Nous noterons aussi que lors d'une contraction, la première colonne dans la liste du réseau n'est pas ajoutée à la liste des voisins, car elle correspond au tenseur qui fait déjà partie du réseau central.

Il est bon de noter que ce code peut aussi accommoder la contraction d'une section du réseau. Pour faire cette contraction, il suffit de trouver les positions des indices à travers lesquels la coupure est faite, et de remplacer leurs voisins par des sorties ouvertes. Le code peut par la suite être utilisé comme décrit auparavant. La capacité du code à permettre ce genre de contraction partielle démontre un usage potentiel important, c'est-à-dire la recherche et le calcul d'isométries à travers le réseau. Comme nous en avons parlé dans la théorie, un intérêt central de ces réseaux de tenseurs est la création de codes de correction d'erreurs, et la présence d'isométries dans le code donne un bon indice quant à la capacité de ce dernier de corriger une erreur sur la frontière.

2.3 Conversion à la matrice de covariance

Cette section couvre la dernière partie de l'algorithme, soit la conversion entre la matrice génératrice obtenue dans la section "Contraction du réseau" et la matrice de covariance, de laquelle nous allons obtenir les observables voulues, telle l'intrication, le "self-energy", et maintes autres. Nous allons écrire explicitement le calcul théorique derrière l'algorithme simultanément avec le code, afin que celui-ci soit compatible avec la notation utilisée. Cette démonstration a déjà été effectuée par Alexander Jahn et al [30], mais leur preuve contient certaines erreurs et imprécisions et donc mérite d'être refaite ici. Quelques-unes des méthodes utilisées dans leur preuve seront également différentes. Chaque partie de la preuve sera accompagnée d'un morceau de code y correspondant. Commençons d'abord la preuve théorique. Nous cherchons à exprimer la matrice de covariance, qui possède la forme suivante :

$$\Gamma_{j,k}(\psi) = \langle\psi| \frac{i}{2}[\gamma_i, \gamma_j] |\psi\rangle \tag{2.15}$$

Afin de calculer cette équation, nous allons partir de la matrice de covariance pour un état vide

$$\Gamma_{j,k}(\varnothing) = \langle\varnothing| \frac{i}{2}[\gamma_i, \gamma_j] |\varnothing\rangle \,, \tag{2.16}$$

qui se calcule aisément en transformant les opérateurs de Majorana en opérateurs de spins et calculant le corrélateur. On obtient une matrice de la forme

$$\Gamma(\varnothing) = \bigoplus_{i=1}^{N} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \tag{2.17}$$

qui par la suite est transformée par deux rotations gaussiennes dans la forme désirée.

```
% Gam0 is the covariant matrix for an empty state
gam0 = [0,1 ;-1,0] ;
Gam0 = gam0 ;
for k = 1 :L - 1
    Gam0 = blkdiag(Gam0,gam0) ;
end
```

La formule résultante est

$$\Gamma(\psi) = P\tilde{W}V_\phi P^{-1}\Gamma(\emptyset)(P\tilde{W}V_\phi P^{-1})^T. \tag{2.18}$$

La clé du développement de ces opérateurs est une transformation de Bogoliubov. Nous commençons par une transformation unitaire appliquée sur les opérateurs de création

$$\tilde{f}_i^\dagger = U_W f_i^\dagger U_W^\dagger = \sum_{j=1}^{N} W_{i,j} f_j^\dagger, \tag{2.19}$$

qui change l'exponentielle de la fonction caractéristique d'une forme gaussienne à une forme normale.

$$\begin{aligned}\sum_{i,j=1}^{N} A_{i,j} f_i^\dagger f_j^\dagger &= \sum_{i,j=1}^{N}\sum_{i',j'=1}^{N} (W^T)_{i,i'}\Sigma_{i,j}W_{j',j} f_i^\dagger f_j^\dagger \\ &= \sum_{i,j=1}^{N}\sum_{i',j'=1}^{N} \Sigma_{i,j}(W_{i',i} f_i^\dagger)(W_{j',j} f_j^\dagger) \\ &= \sum_{i',j'=1}^{N} \Sigma_{i,j}\tilde{f}_{i'}^\dagger \tilde{f}_{j'}^\dagger\end{aligned} \tag{2.20}$$

Ici nous avons utilisé une propriété des matrices antisymétriques N x N, qui peuvent être décrites via un changement de base par une matrice diagonale par bloc de 2 x 2, avec un bloc de 0 ajouté si N est impair. L'unitaire W ici est également de taille N.

$$A = W^T\Sigma W \to \Sigma = \frac{1}{2}\bigoplus_{i=1}^{N/2}\begin{pmatrix} 0 & \lambda_i \\ -\lambda_i & 0 \end{pmatrix} \tag{2.21}$$

Nous calculons la matrice W dans notre algorithme en diagonalisant la matrice $A0$ prise en entrée, appliquant par la suite la transformation $\Sigma = JDJ^t$ sur la matrice diagonale D résultante, avec

$$J = \bigoplus_{i=1}^{N/2}\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ i & -i \end{pmatrix}. \tag{2.22}$$

On ajoute 1 si la matrice est de taille impaire. La matrice W est ensuite calculée comme le produit de la matrice J et la matrice de changement de base U, qui est composée des vecteurs propres de A.

```
function GamFin = gentocorr2(A0)
\% generator matrix to correlation matrix
L = size(A0,1) ;
[U,D] = eig(A0) ;
[~,vvv] = sort(abs(diag(D) + 1e-10),'descend') ; % Sorts the eigenvalues so that
    each positive eigenvalue is followed by the corresponding negative eigenvalue
D = D(vvv,vvv) ;
U = U( :,vvv) ;
% We use J to transform the diagonal matrix into a block diagonal matrix
X = [1 1 ; 1i -1i]/sqrt(2) ;
J = kron(eye(floor(L/2)),X) ;
if mod(L,2) == 1
    J = blkdiag(J,1) ;
end
E = J*D*J' ;
W = J*U' ;
```

Exprimant l'équation 2.20 en substituant la matrice Σ, on obtient

$$\sum_{i,j=1}^{N} A_{i,j} f_i^\dagger f_j^\dagger = \sum_{i'=1}^{N/2} \lambda_{i'} \tilde{f}_{2i'-1}^\dagger \tilde{f}_{2i'}^\dagger, \tag{2.23}$$

qui permet de découpler l'exponentielle dans la fonction caractéristique d'une porte de parité de la manière suivante

$$\begin{aligned} |\psi\rangle &= c \exp\left(\sum_{i,j=1}^{N} A_{i,j} f_i^\dagger f_j^\dagger\right)|\varnothing\rangle = c \exp\left(\sum_{i'=1}^{N/2} \lambda_{i'} \tilde{f}_{2i'-1}^\dagger \tilde{f}_{2i'}^\dagger\right)|\varnothing\rangle \\ &= c \prod_{i=1}^{N/2} \exp\left(\lambda_{i'} \tilde{f}_{2i'-1}^\dagger \tilde{f}_{2i'}^\dagger\right)|\varnothing\rangle = c \prod_{i=1}^{N/2} \left(1 + \lambda_{i'} \tilde{f}_{2i'-1}^\dagger \tilde{f}_{2i'}^\dagger\right)|\varnothing\rangle . \end{aligned} \tag{2.24}$$

La dernière ligne est due à la propriété des variables de Grassmann $f^{\dagger 2} = 0$. On normalise maintenant la fonction d'onde.

$$\langle\psi|\psi\rangle = c^2 \prod_{i=1}^{N/2} \left(1+\lambda_{i'}^2\right) = 1 \rightarrow c = \left(\prod_{i=1}^{N/2} \left(1+\lambda_{i'}^2\right)\right)^{-1/2}, \tag{2.25}$$

ce qui permet de découpler la fonction d'onde, chaque terme possédant un i agissant comme un opérateur unitaire agissant sur $|\emptyset\rangle$.

Appliquant la relation 2.19 sur les opérateurs de Majorana, on retrouve la relation $\tilde{\Gamma}(\psi) = \tilde{W}\Gamma(\psi)\tilde{W}^t$, avec

$$\tilde{W} = W \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{2.26}$$

Cette transformation de l'opérateur permet d'accommoder la dimension de l'espace des opérateurs de Majorana, qui est de $2N$, et prend en compte que l'opérateur agissant sur $\tilde{f}_i^\dagger$ agit sur γ_{2i-1} et sur γ_{2i}.

```
% Transforms W into W x I
Wtilda = zeros(2*L,2*L) ;
for k = 1 :L
    for p = 1 :L
        Wtilda(2*k-1 :2*k,2*p-1 :2*p) = (W(k,p)*eye(2)) ;
    end
end
```

Une fois la matrice de covariance dans la base appropriée, nous effectuons la deuxième rotation gaussienne, permettant de passer de $\tilde{\Gamma}(\emptyset)$ à $\tilde{\Gamma}(\psi)$. Pour ceci nous partons de la fonction caractéristique, que nous transformons en une série d'opérateurs. On rappelle que dans l'équation 2.25 nous avions conclu que la fonction caractéristique pouvait être construite de la manière suivante :

$$T = \prod_{i=1}^{N/2} \frac{1+\lambda_{i'}\tilde{f}_{2i'-1}^\dagger \tilde{f}_{2i'}^\dagger}{\sqrt{1+\lambda_{i'}^2}}. \tag{2.27}$$

Nous allons définir les variables

$$\cos\theta_i = \frac{1}{\sqrt{1+\lambda_i^2}}, \quad \sin\theta_i = \frac{\lambda_i}{\sqrt{1+\lambda_i^2}}, \tag{2.28}$$

et $\Omega_i = \tilde{f}^\dagger_{2i'-1}\tilde{f}^\dagger_{2i'} + \tilde{f}_{2i'-1}\tilde{f}_{2i'}$, qui permettent de formuler la transformation unitaire

$$\frac{1+\lambda_{i'}\tilde{f}^\dagger_{2i'-1}\tilde{f}^\dagger_{2i'}}{\sqrt{1+\lambda_{i'}^2}} = I\cos\theta_i + \Omega_i \sin\theta_i = e^{\theta_i\Omega_i}. \tag{2.29}$$

```
% Calculates the list of angles
lambdatemp = abs(diag(D)) ;
lambdakeep = lambdatemp(1 :2 :end) ;
thet = acos(1./sqrt(1+lambdakeep.^2)) ;
```

Utilisant le fait que les Ω_i commutent, on arrive à l'opérateur unitaire $U_\theta = \exp\left\{\sum_{i=1}^{L/2}\theta_i\Omega_i\right\}$. Cet opérateur, combiné avec les équations de mouvement $\tilde{f}_i^\dagger(\theta) = U_\theta^\dagger \tilde{f}_i^\dagger U_\theta$, donne la rotation globale

$$V_\theta = \bigoplus_{i=1}^{N/2}\left(I_4\cos\theta_i + \sigma_y\otimes\sigma_x\sin\theta_i\right), \tag{2.30}$$

où nous ajoutons I_2 si N est impair.

```
% Calculate Gamtemp, which acts as V
k = 1 ;
ct = cos(thet(k)) ; st = sin(thet(k)) ;
Gamtemp = [ct,0,0,st ;0,ct,st,0 ;0,-st,ct,0 ;-st,0,0,ct] ;
for k = 2 :L/2
    ct = cos(thet(k)) ; st = sin(thet(k)) ;
    Gamtemp = blkdiag(Gamtemp,[ct,0,0,st ;0,ct,st,0 ;0,-st,ct,0 ;-st,0,0,ct]) ;
end
% Add I if L is odd
if mod(L,2) == 1
    Gamtemp = blkdiag(Gamtemp,eye(2)) ;
end
```

Considérant que cet opérateur ainsi que $\tilde{W}$ sont dans la base des opérateurs de création,

nous effectuons le changement de base

$$P = \bigoplus_{i=1}^{N} \begin{pmatrix} 1 & 1 \\ i & -i \end{pmatrix}, \tag{2.31}$$

ce qui donne finalement le résultat voulu, l'équation 2.18.

```
% Create Pmat, which acts as P
pmat = [1,1 ;-1i,1i]/sqrt(2) ;
Pmat = pmat ;
for k = 1 :L - 1
    Pmat = blkdiag(Pmat,pmat) ;
end

% Calculate Gamma (GamFin)
Gamtot = Pmat*Wtilda'*Gamtemp*Pmat' ;
GamFin = Gamtot*Gam0*transpose(Gamtot) ;
```

2.4 Résultats obtenus

Afin de confirmer la validité de l'ensemble du code, nous allons effectuer deux comparaisons graphiques avec les résultats obtenus dans l'article de référence de Jahn et al. [30]. La première comparaison est faite avec la figure 3 dans l'article, une figure qui illustre les matrices de covariance pour un réseau {5,4} de distance 1 à trois stades différents de construction. Le réseau utilise la matrice génératrice

$$A = \begin{bmatrix} 0 & -1 & 1 & 1 & -1 \\ 1 & 0 & -1 & 1 & 1 \\ -1 & 1 & 0 & -1 & 1 \\ -1 & -1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 1 & 0 \end{bmatrix}.$$

Le résultat obtenu, pour les trois réseaux, se retrouve dans la figure 2.12. Chacun des réseaux permet de tester une partie différente du code. Le réseau constituant un tenseur est le test de l'algorithme *gentocorr2*, qui convertit la matrice génératrice en la matrice de covariance. Le deuxième réseau, celui du centre dans la figure 2.12, permet de tester les fonctions de contraction *tcontract* et de décalage *cycleperm*. Le dernier graphique du réseau complet permet de tester l'algorithme d'autocontraction *selfcontract*, et l'algorithme complet incluant *createcirclist* et ses sous-fonctions ainsi que *Contractnet*.

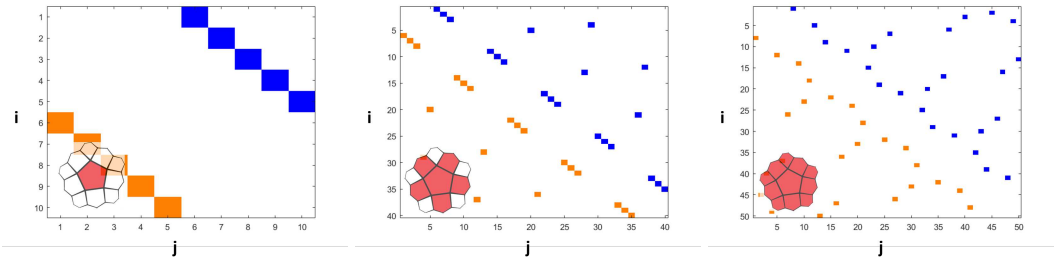


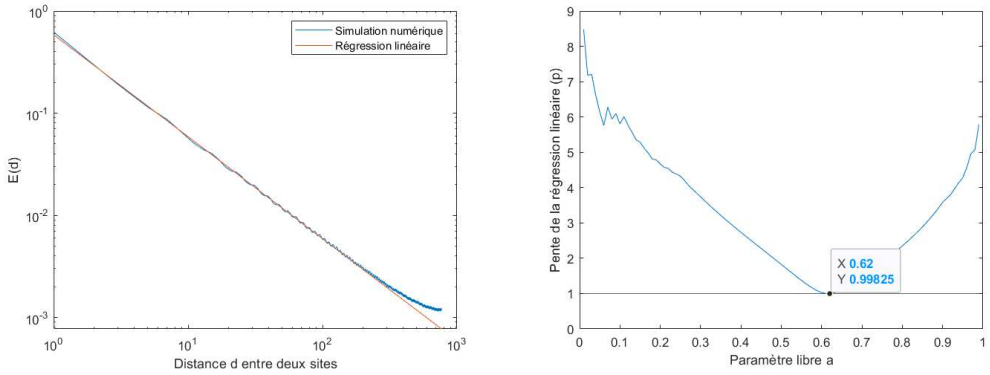
FIGURE 2.12 Trois matrices de covariance obtenues à des stades différents lors de la construction d'un réseau {5,4} de distance 1 avec l'algorithme *concentrique*. Les dimensions des deux axes sont égales au nombre de modes de Majorana dans l'état correspondant au réseau, soit deux fois le nombre de modes fermioniques. Les valeurs en bleu sont égales à +1, et les valeurs en orange équivalent à -1. Le réseau correspondant à la matrice de covariance est illustré en rouge dans le coin inférieur gauche de chaque graphique.

Comme nous pouvons le voir dans la figure 2.12, les trois matrices de covariance correspondent à celles de la figure 3 dans [30]. Nous pouvons donc confirmer que le code fonctionne comme attendu dans le contexte de ce réseau.

La deuxième comparaison est effectuée avec une partie de la figure 5 dans le même article. Dans cette figure, nous ne regardons que les graphiques traitant du réseau {3,7}. Cette deuxième comparaison regarde la diminution de la corrélation E_d entre deux sites en fonction de l'augmentation de la distance d entre eux. Cette figure a été obtenue par Jahn et al. utilisant une méthode de croissance géodésique pour un réseau {3,7}, tandis que nous utilisons la méthode de croissance concentrique pour un réseau {3,7}. Nous voulons donc confirmer que l'algorithme donne un taux de diminution qualitativement similaire. Nous recherchons donc une diminution de E_d avec une forme d^{-p}, où p est une fonction du paramètre libre a dans la matrice génératrice. Nous voulons aussi une valeur critique du paramètre a entre 0 et 1. Les résultats de cette simulation se retrouvent dans la figure 2.13. La formule qui calcule E_d est

$$E_d = \sum_{k=1}^{N} \frac{|\Gamma_{k,k+d}|}{N}, \tag{2.32}$$

avec $\Gamma_{k,j+N} = \Gamma_{k,j}$.



(a) (b)

FIGURE 2.13 La figure (a) illustre la diminution de E_d en fonction de la distance pour un réseau {3,7} de distance 5 créé avec l'algorithme *concentrique* avec un paramètre $a = 0.62$. La pente p de la régression linéaire est 0.99823. La figure (b) est le graphique des valeurs de p pour $0.01 < a < 0.99$. On voit qu'il existe un point critique pour $a = 0.62$.

On voit que la diminution de E_d en d^{-p} est en effet présente, et indique la présence d'une théorie de champ d'Ising, comme le confirme l'article de référence. La valeur critique du paramètre a est de 0.62, ce qui est proche de la valeur obtenue par Jahn et al d'environ 0.6, et correspond à l'état fondamental du hamiltonien

$$H = i\left(\sum_{k=1}^{N-1} \gamma_k \gamma_{k+1} + \gamma_1 \gamma_N\right). \tag{2.33}$$

Nous pouvons confirmer que l'utilisation d'une distance non-géodésique ne retourne pas exactement les mêmes résultats, mais suit le comportement attendu et retourne des valeurs critiques qui sont suffisamment proches pour être utilisées en tant qu'approximation.

Bien que nous avons pû confirmer la présence d'une théorie de champ d'Ising, faire des études plus approfondies sur l'état au périmètre ne serait pas particulièrement utile, considérant que les tenseurs existent à différentes échelles d'énergie de la CFT. Nous pouvons cependant voir des tendances. Un exemple de cela peut être observé dans la figure 2.14, où nous avons calculé la valeur moyenne E de la corrélation entre deux densités d'énergie ϵ en

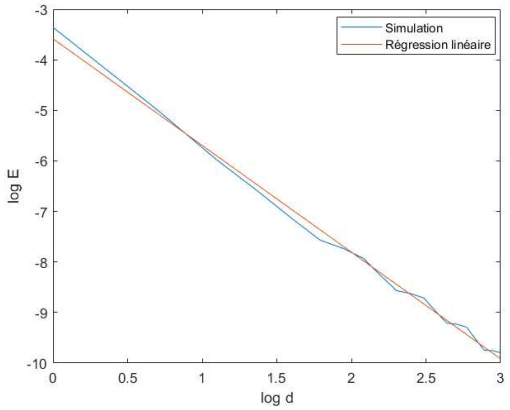


FIGURE 2.14 La diminution de la corrélation E entre la densité d'énergie de deux points en fonction de la distance d séparant ces points pour un réseau {3,7} de distance 4. La pente rouge est la régression linéaire calculée et a une valeur de -2.1104.

fonction de la distance entre deux points en utilisant la formule

$$E = \frac{2}{N}\sum_{i=1}^{N/2}\Big(\langle\epsilon_i\epsilon_{i+d}\rangle - \langle\epsilon_i\rangle\langle\epsilon_{i+d}\rangle\Big) = \frac{1}{2N}\sum_{i=1}^{N/2}\Gamma_{2i-1,2i+2d}\Gamma_{2i,2i+2d-1}. \tag{2.34}$$

Une régression linéaire de $log(E)$ en fonction de $log(d)$, où d est la distance entre les deux sites, donne l'équation

$$E \sim d^{-2.1104}. \tag{2.35}$$

Comparant cette valeur à celle retrouvée dans la figure 14 de l'article de référence [30], qui est de -2.0243, nous voyons que les deux valeurs sont relativement proches. Donc bien que nous ne pouvons pas utiliser notre code pour retrouver les valeurs exactes, il est possible d'évaluer qualitativement le comportement des corrélations sur l'état à la frontière en utilisant le code.

Conclusion

Nous avons introduit un algorithme qui permet de créer un réseau de tenseurs avec un dallage et une profondeur arbitraires ainsi que d'obtenir la matrice de covariance associée au réseau. Ce code peut également supporter tout état pair. En regardant le réseau résultant de ce code, nous voyons qu'il répond à plusieurs des critères qui ont été donnés dans l'introduction. Le code est efficace et performe sans trop de ressources, créant un tenseur à plusieurs centaines, voire quelques milliers de sites sur un ordinateur portable commercial. Le réseau de tenseurs créé peut être hyperbolique si le bon dallage est spécifié. Étant donné que chaque tenseur est le même, le réseau est également isotrope. Avec le fait que l'état sur le périmètre du réseau possède des corrélateurs qui diminuent algébriquement en fonction de la distance, avec une valeur critique correspondant à une CFT d'Ising, il est raisonnable de considérer que le code crée des réseaux qui ont un potentiel de représenter la correspondance AdS/CFT. Bien que ces résultats aient déjà été atteints par l'article de référence écrit par Jahn et al. [30], nous avons en plus bonifié le contenu de cet article pour le rendre accessible à tous.

Pendant que nous décrivions le code, nous avons pu corriger certaines erreurs et omissions dans l'article de référence. La description mathématique de la conversion entre matrice génératrice et matrice de covariance a été faite avec une notation constante à travers la sous-section, et cette cohérence dans la notation utilisée est maintenue à partir de la théorie. D'autres coquilles ont également été corrigées dans la démonstration, la rendant plus accessible. Nous avons également donné un algorithme explicite pour transformer la matrice génératrice de forme antisymétrique à la forme normale, ce qui était absent dans l'article de référence. Lors de l'explication sur l'auto-contraction, la possibilité d'obtenir des valeurs divergentes dans la matrice génératrice en utilisant $a = 1$ a été mentionnée, ce qui était absent dans les instructions de l'article de référence. Finalement, nous avons discuté d'un algorithme de rotation différent de celui présent dans l'article de référence. Ce protocole, qui effectue une rotation dans la base de spin et non dans la base fermionique, permet de contracter le réseau sans prévoir chaque étape de la contraction. Cette contrainte, qui

n'a pas été mentionnée dans l'article de référence, est cruciale, car l'application des algorithmes présents dans l'article de référence sans connaissance préalable mène à des résultats invalides.

Il reste néanmoins plusieurs améliorations et tests à effectuer pour que nous puissions confirmer la capacité des réseaux créés par ce code à représenter la correspondance AdS/CFT. La première modification serait d'implémenter les opérations de base (la permutation cyclique, la contraction et l'auto-contraction) pour les tenseurs impairs. La suivante est d'ajouter l'algorithme de croissance géodésique au répertoire, afin de pouvoir éviter la coexistence de deux niveaux d'énergie distincts sur la frontière d'un réseau. La prochaine étape est de développer un programme qui peut afficher les réseaux de tenseurs à partir de la matrice créée par *createcirclist*. Cette étape serait utile, permettant à l'utilisateur du code de visualiser le réseau généré et de s'apercevoir de la présence d'erreurs plus facilement. Un tel code n'a pas été développé ici par contrainte de temps, mais aurait dû être une priorité lors du début du projet. Finalement, il serait nécessaire d'amender le code pour permettre l'ajout de tenseurs intermédiaires qui seraient situés entre chaque tenseur du dallage. Ceci permettrait d'augmenter les paramètres libres du système, et ainsi la formation plus aisée d'agencements qui pourraient constituer une isométrie. Pour ce faire avec le code actuel, nous proposons d'inclure l'ajout de ces tenseurs soit lors de la contraction entre deux tenseurs, ou d'un tenseur avec lui-même, ou d'ajouter une étape après chaque itération d'un des algorithmes de croissance.

Nous concluons ce mémoire en regardant les ouvertures offertes par les réseaux de tenseurs hyperboliques, ainsi que les portes de parité elles-mêmes. Un concept que nous n'avons pas abordé est la relation entre le dallage et la métrique même de l'espace dans lequel nous travaillons. Par exemple, les dallage {3,7} et {5,4} ont été utilisés interchangeablement lors du mémoire pour spécifier le même espace-temps, mais cela n'est pas nécessairement le cas. Certaines sources ont spéculé qu'un dallage représente directement la métrique de l'espace-temps [54][55]. Si cela est vrai, une question intéressante émerge. Est-ce que les dallages permettant une CFT non-triviale sur le périmètre peuvent correspondre à une métrique physique, ou sommes-nous contraints à l'espace de Minkowski ?

Bibliographie

[1] polytope24. (2+1)-dimensional anti-de sitter space. *https://commons.wikimedia.org/*, 2013. [En ligne, sous licence CC BY-SA 3.0].

[2] Glen Evenbly. Hyperinvariant tensor networks and holography. *Physical review letters*, 119(14) :141602, 2017.

[3] Alexei Kitaev and Chris Laumann. Topological phases and quantum computation. *Exact methods in low-dimensional statistical physics and quantum computing," Lecture Notes of the Les Houches Summer School*, (89) :101–125, 2009.

[4] R Walter Ogburn and John Preskill. Topological quantum computation. In *NASA International Conference on Quantum Computing and Quantum Communications*, pages 341–356. Springer, 1998.

[5] P Jouzdani, E Novais, and ER Mucciolo. Fidelity of the surface code in the presence of a bosonic bath. *Physical Review A*, 88(1) :012336, 2013.

[6] Simon J Devitt, William J Munro, and Kae Nemoto. Quantum error correction for beginners. *Reports on Progress in Physics*, 76(7) :076001, 2013.

[7] Barbara M Terhal. Quantum error correction for quantum memories. *Reviews of Modern Physics*, 87(2) :307, 2015.

[8] Daniel Gottesman. Stabilizer codes and quantum error correction. *arXiv preprint quant-ph/9705052*, 1997.

[9] Emanuel Knill and Raymond Laflamme. Theory of quantum error-correcting codes. *Physical Review A*, 55(2) :900, 1997.

[10] A Robert Calderbank and Peter W Shor. Good quantum error-correcting codes exist. *Physical Review A*, 54(2) :1098, 1996.

[11] David Kribs, Raymond Laflamme, and David Poulin. Unified and generalized approach to quantum error correction. *Physical review letters*, 94(18) :180501, 2005.

[12] Benjamin J Brown, Naomi H Nickerson, and Dan E Browne. Fault-tolerant error correction with the gauge color code. *Nature communications*, 7(1) :1–8, 2016.

[13] Nikolas P Breuckmann, Kasper Duivenvoorden, Dominik Michels, and Barbara M Terhal. Local decoders for the 2d and 4d toric code. *arXiv preprint arXiv:1609.00510*, 2016.

[14] Albert H Werner, Daniel Jaschke, Pietro Silvi, Martin Kliesch, Tommaso Calarco, Jens Eisert, and Simone Montangero. Positive tensor network approach for simulating open quantum many-body systems. *Physical review letters*, 116(23) :237201, 2016.

[15] Luca Tagliacozzo, Alessio Celi, and Maciej Lewenstein. Tensor networks for lattice gauge theories with continuous groups. *Physical Review X*, 4(4) :041024, 2014.

[16] Glen Evenbly and Guifre Vidal. Tensor network renormalization. *Physical review letters*, 115(18) :180405, 2015.

[17] Norbert Schuch. Condensed matter applications of entanglement theory. *arXiv preprint arXiv:1306.5551*, 2013.

[18] Pietro Silvi, Enrique Rico, Tommaso Calarco, and Simone Montangero. Lattice gauge tensor networks. *New Journal of Physics*, 16(10) :103015, 2014.

[19] Román Orús. Tensor networks for complex quantum systems. *Nature Reviews Physics*, 1(9) :538–550, 2019.

[20] Thomas Huckle, Konrad Waldherr, and Thomas Schulte-Herbrüggen. Computations in quantum tensor networks. *Linear Algebra and its Applications*, 438(2) :750–781, 2013.

[21] Christopher J Wood, Jacob D Biamonte, and David G Cory. Tensor networks and graphical calculus for open quantum systems. *arXiv preprint arXiv:1111.6950*, 2011.

[22] Edwin Stoudenmire and David J Schwab. Supervised learning with tensor networks. In *Advances in Neural Information Processing Systems*, pages 4799–4807, 2016.

[23] William Huggins, Piyush Patil, Bradley Mitchell, K Birgitta Whaley, and E Miles Stoudenmire. Towards quantum machine learning with tensor networks. *Quantum Science and technology*, 4(2) :024001, 2019.

[24] Luciano Serafini and Artur S d'Avila Garcez. Learning and reasoning with logic tensor networks. In *Conference of the Italian Association for Artificial Intelligence*, pages 334–348. Springer, 2016.

[25] Ivan Glasser, Nicola Pancotti, and J Ignacio Cirac. Supervised learning with generalized tensor networks. *arXiv preprint arXiv:1806.05964*, 2018.

[26] Igor L Markov and Yaoyun Shi. Simulating quantum computation by contracting tensor networks. *SIAM Journal on Computing*, 38(3) :963–981, 2008.

[27] Jacob Biamonte and Ville Bergholm. Tensor networks in a nutshell. *arXiv preprint arXiv:1708.00006*, 2017.

[28] Ahmed Almheiri, Xi Dong, and Daniel Harlow. Bulk locality and quantum error correction in ads/cft. *Journal of High Energy Physics*, 2015(4) :163, 2015.

[29] Fernando Pastawski, Beni Yoshida, Daniel Harlow, and John Preskill. Holographic quantum error-correcting codes : Toy models for the bulk/boundary correspondence. *Journal of High Energy Physics*, 2015(6) :149, 2015.

[30] Alexander Jahn, Marek Gluza, Fernando Pastawski, and Jens Eisert. Holography and criticality in matchgate tensor networks. *Science advances*, 5(8) :eaaw0092, 2019.

[31] Leslie G Valiant. Expressiveness of matchgates. *Theoretical Computer Science*, 289(1) :457–471, 2002.

[32] Sergey Bravyi. Contraction of matchgate tensor networks on non-planar graphs. *Contemp. Math*, 482 :179–211, 2009.

[33] Hong-Chen Jiang, Zheng-Yu Weng, and Tao Xiang. Accurate determination of tensor network state of quantum lattice models in two dimensions. *Physical review letters*, 101(9) :090603, 2008.

[34] Wei Li, Shi-Ju Ran, Shou-Shu Gong, Yang Zhao, Bin Xi, Fei Ye, and Gang Su. Linearized tensor renormalization group algorithm for the calculation of thermodynamic properties of quantum lattice models. *Physical review letters*, 106(12) :127202, 2011.

[35] Luciano Serafini and Artur d'Avila Garcez. Logic tensor networks : Deep learning and logical reasoning from data and knowledge. *arXiv preprint arXiv:1606.04422*, 2016.

[36] Andrew J Ferris and David Poulin. Tensor networks and quantum error correction. *Physical review letters*, 113(3) :030501, 2014.

[37] Andrew Darmawan, Pavithran Iyer, and David Poulin. Critical noise parameters for fault-tolerant quantum computation. 2016.

[38] Guiffre Vidal. Tensor network renormalization - g. vidal - 2/24/2015. *https://www.youtube.com/watch?v=iAvpqJ7FqHA*, 2015.

[39] John S Townsend. *A modern approach to quantum mechanics*. University Science Books, 2000.

[40] Thomas E Baker, Samuel Desrosiers, Maxime Tremblay, and Martin P Thompson. Méthodes de calcul avec réseaux de tenseurs en physique (basic tensor network computations in physics). *arXiv preprint arXiv:1911.11566*, 2019.

[41] Gary W Gibbons. Anti-de-sitter spacetime and its uses. In *Mathematical and quantum aspects of relativity and cosmology*, pages 102–142. Springer, 2000.

[42] Mark Van Raamsdonk. Building up spacetime with quantum entanglement. *General Relativity and Gravitation*, 42(10) :2323–2329, 2010.

[43] Lopez Rafael. Differential geometry of curves and surfaces in lorentz-minkowski space. *Int. Electron. J. Geom.*, 7, 11 2008.

[44] Alexander Jahn, Marek Gluza, Fernando Pastawski, and Jens Eisert. Majorana dimers and holographic quantum error-correcting codes. *Physical Review Research*, 1(3) :033079, 2019.

[45] Brayden Ware, Jun Ho Son, Meng Cheng, Ryan V Mishmash, Jason Alicea, and Bela Bauer. Ising anyons in frustration-free majorana-dimer models. *Physical Review B*, 94(11) :115127, 2016.

[46] Ying Tang, Anders W Sandvik, and Christopher L Henley. Properties of resonating-valence-bond spin liquids and critical dimer models. *Physical Review B*, 84(17) :174427, 2011.

[47] Roderich Moessner, Shivaji L Sondhi, and Eduardo Fradkin. Short-ranged resonating valence bond physics, quantum dimer models, and ising gauge theories. *Physical Review B*, 65(2) :024504, 2001.

[48] G Kells, JK Slingerland, and Jiri Vala. Description of kitaev's honeycomb model with toric-code stabilizers. *Physical Review B*, 80(12) :125415, 2009.

[49] Emanuel Knill. Fermionic linear optics and matchgates. *arXiv preprint quant-ph/0108033*, 2001.

[50] Richard Jozsa and Akimasa Miyake. Matchgates and classical simulation of quantum circuits. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 464(2100) :3089–3106, 2008.

[51] Charles HC Little. Kasteleyn's theorem and arbitrary graphs. *Canadian Journal of Mathematics*, 25(4) :758–764, 1973.

[52] Gianfranco Cariolaro and Gianfranco Pierobon. Reexamination of bloch-messiah reduction. *Physical Review A*, 93(6) :062115, 2016.

[53] Adrian Franco Rubio. Holographic quantum error correcting codes. 2016. http ://www.afrancorubio.com/TFGMat.pdf.

[54] Tamara Kohler and Toby Cubitt. Complete toy models of holographic duality. *arXiv preprint arXiv:1810.08992*, 2018.

[55] Patrick Hayden, Sepehr Nezami, Xiao-Liang Qi, Nathaniel Thomas, Michael Walter, and Zhao Yang. Holographic duality from random tensor networks. *Journal of High Energy Physics*, 2016(11) :9, 2016.